Digitized by the Internet Archive
in 2026 with funding from
Brewster Kahle

https://archive.org/details/terrapininc00unse

Brewster wrote this

TASMAN TURTLE MANUAL

written by

Terrapin and Flexible Systems

Terrapin Inc.
380 Green Street
Cambridge MA 02139

Introduction

The Tasman Turtle is an accurate versatile robot that is easy to command from any computer language. Logo routines have been written (and are included) to help you start. With your turtle you can draw complex figures, and "feel" your way around a room.

The Turtle is interesting and educational for youngsters and researchers alike. Hook it up and try it.

Setup

Your Tasman Turtle comes with:

Power Supply (black box)

Interface Board and Cable

Tasman Turtle Control Language Disk

Warranty Card

Please send in the warranty card to get information on changes and enhancements to your turtle. This is an important step to maintain support for your turtle.

To use your turtle:

1) Turn off your Apple

2) Plug the interface card into **slot 2**

3) Plug in the two banana plugs into your Turtle Power Supply box. Red goes in red and black in black.

4) Turn off the power supply

5) Plug your power supply into the wall.

Now you can turn on your Apple and load in your programs. If you turn on your Turtle Power Supply before running the SETUP procedure (to put the turtle in an initial state) the turtle may have its eyes on or the horn on. This may be annoying, but not harmful.

Two small red lights should be on inside the turtle on the green boards. Further, if you push the touch ring against a sensor switch, the corresponding lamp should light in the group of four lights in the front. If none of these lights work, go back and check to see that the power supply is hooked up correctly.

Logo Interface

The Tasman Turtle can be commanded directly from Terrapin Logo by loading the routines on the accompanying disk in the file "TASTCL". You should feel free to modify the enclosed programs, or add new ones to extend your turtle's capability.

Here is a summary of the command you can use immediately

SETUP should be run after reading "TASTCL"

TFD :dist movement commands

TBK :dist

TRT :angle

TLT :angle

EYESON eyes commands

EYESOFF

TPU turtle pen commands

TPD

HORNHI horn commands

HORNLO

HORNOFF

MOVEUNTIL :direction :distance :condition (see conditional motion section)

LTOUCH? outputs the state of the corresponding touch

sensor- true or false. ex: IF LTOUCH? TLT 30

RTOUCH?

FTOUCH?

BTOUCH?

FTOUCHONLY?

BTOUCHONLY?

LTOUCHONLY?

RTOUCHONLY?

ANYTOUCH?

NOTOUCH?

TOUCH outputs the state of the touch sensors as a number computed as follows:
start with 0 and then
if the front is touched then add :FRONTBIT (4)
if the back is touched then add :BACKBIT (8)
if the left is touched then add :LEFTBIT (1)
if the right is touched then add :RIGHTBIT (2)

Most of these are discussed in the Terrapin Tutorial so I will only discuss the ones that are different.

Motion Commands

The motion commands (eg. TFD) are executed by a piece of assembly code for smooth motion. The source code is F.CODE and in the Logo Assembler format, allowing you to add or modify these routines. (see the Terrapin Technical Manual for Assembler information) The assembled code is in the file TASTCL.BIN and is automatically loaded by setup. IF you have some special code that you keep in the machine language space of Logo then you can use the routines in "TTCL" instead of "TASTCL"; but this will not bother most people).

Conditional Motion

The Tasman Turtle has the ability to move until it hits something. Before each turtle step, whether turning or going straight, the turtle can test its bumpers to see if it should proceed. This allows the user to control what the turtle should do if it hits something along the way.

The easiest way to use this feature is the MOVEUNTIL procedure.

MOVEUNTIL :direction :distance :condition

If you called MOVEUNTIL TFD 100 :FRONTBIT, the turtle will move 100 turtle steps,

unless it hits something in front. In that case is would stop moving and go on to the next command. The turtle can be made sensitive to more than one direction by adding the BITS before the MOVEUNTIL command.

eg. MOVEUNTIL TFD 100 :FRONTBIT + :LEFTBIT

will be sensitive to front and left touches but will ignore back and right touches.

FOR EXPERTS ONLY

Another way to use this feature is the UNTIL command.

UNTIL :condition

Takes a number or word as input and does that test before **all steps in the future.** So if you said UNTIL :FRONTBIT and then TFD 100 it would do the same as MOVEUNTIL example but the condition would persist in future motion. If you call UNTIL 0 the turtle will not stop for anything, and if you call UNTIL :ANYBIT it will stop for anything.

Calling UNTIL with a number is a versatile way to control the conditions. Combination of conditions is made by adding them as in MOVEUNTIL

The only procedures that change the UNTIL set condition, is SETUP and MOVEUNTIL.

I recommend that you play with it to see what it does. You might want to change the UNTIL procedure.

Screenturtle Option

You can have the turtle's movements reflected by the screen turtle by typing:

MAKE "SCREENTURTLE "TRUE

Many people like this option. If you want to happen all of the time you can change the SETUP routine to make "screenturtle "true rather than "false as the routine is now written.

Example

After hooking up your turtle and loading Terrapin Logo type:

READ "TASTCL

SETUP now you are ready to go.

Maybe you want to make a square, so type

REPEAT 4 [TFD 20 TRT 90] and away it should go.

Fine Adjustments

Certain parameters can be changed to suit your preference.

:CALIBR and :CALIBL are calibration variables to make your turtle work exactly in degrees. This may range from 1.0 to 1.1 (default is 1.05). You may want to change this if your turtle does not turn exactly 180 degrees. :CALIBR is for right adjustments and :CALIBL is for left adjustments.

TSTEP is the size of each turtle step in the Forward and Backward routines. This can be changed by

.DEPOSIT :TSTEP <number> (default it 10).

TSTALL controls the speed of the turtle. This can be changed a bit but not much. Change by

.DEPOSIT :STALL <number> (default 12).

FLEXIBLE SYSTEMS
INC.
TASMAN TURTLE CO.

219 LIVERPOOL STREET
HOBART. TASMANIA 7000 AUST
Tel. (002) 34 3064

RUNNING YOUR TASMAN TURTLE IN BASIC

The Tasman Turtle is a general purpose personal robot designed to have all the flexibility needed to encourage hobby, educational or professional use. The Turtle is easiest to use when used in conjunction with the higher level language LOGO, however, any other language including machine code can be used to run the Turtle.

The following is a brief guide to allow the Tasman Turtle to be used with Basic. You will discover new possibilities and greater versatility using the turtle robot this way.

Housed inside the Tasman Turtle™ is a General Purpose Interface board which allows up to 64 bits of input/output data to be utilised.

This is arranged as four 8 bit input ports and four 8 bit output ports:

	IN	OUT
PORT 1	(W)	(W)
PORT 2	(W + 1)	(W + 1)
PORT 3	(W + 2)	(W + 2) = R
PORT 4	(W + 3)	(W + 3)

The 8 bits of Port 1 input are dedicated to the 8 Turtle functions described below.

The first four bits of Port 3 output are dedicated to the four Turtle sense switches. All other bits are free for custom use, or for use by Turtle options such as the Turtle Talk ™.

If port 2 of the Apple II is used then W = -16224 and R = -16222 where W stands for Write (for the Turtle) and R stands for Read (from the Turtle)

To operate the turtle functions

POKE W, X

where X = 16 - lamps on
32 - pen down
64 - toot low
192 - toot high

More than one function can operate simultaneously. e.g. to put the pen down and the lamps on

POKE W, 48 (48 = 16 + 32)

To turn everything off

POKE W, Ø

To move the Turtle

```
1ØØ POKE W, A
11Ø GOSUB 1ØØØ
12Ø POKE W, B
13Ø GOSUB 1ØØØ
14Ø GOTO 1ØØ

1ØØØ FOR T = 1 TO TT
1Ø1Ø NEXT T
1Ø2Ø RETURN
```

Where

MOTION		A	B
FORWARD		8	13
BACK		2	7
LEFT		1Ø	15
RIGHT		Ø	5
RIGHT WHEEL		Ø	1
	or	Ø	4
LEFT WHEEL		2	3
	or	8	12

The value of TT is best between Ø - 5; the larger the value the slower the turtle will move.

To sense the touch switches

```
1ØØ A = PEEK (R)
11Ø IF A = X THEN GOSUB
12Ø GOTO 1ØØ
```

or

```
1ØØ A = PEEK (R)
11Ø IF A = 255 THEN 1ØØ
12Ø IF X = 254 THEN
13Ø IF X = 253 THEN
      etc.
```

Where X =

	L	R	F	B
L	254			
R	252	253		
F	25Ø	249	251	
B	246	245	243	247

LFB - 242
RFB - 241
LFR - 248
LBR - 244
LBFR - 24Ø

If no switch is operated then X = 255

TASMAN TURTLE

MINIMUM CONFIGURATION (Minimum Turtle)

Functions:

Left Motor Set Direction
Left Motor Toggle
Right Motor Set Direction
Right Motor Toggle
Horn
Horn Tone
Lights
Pen Solenoid
Auxiliary Input
Left Sense Switch
Right Sense Switch
Front Sense Switch
Back Sense Switch

Specifications:

Power Requirements

12VDC (Nominal)
1.5 Amp (Max.)

Stepping Motors

Step Angle - 7.5°
Voltage - 12VDC
Max. Frequency - 100Hz

Control Inputs

High - On
Low - Off

Sense Switches

Normally - High
Active - Low

Auxiliary Input

12VDC, 1 Amp Drive Current (Max)
High - On
Low - Off

Dimensions

Diameter - 385mm
Height - 180mm
Ground Clearance - 20mm

Drive Mechanism

Direct Drive

Wheel diameter - 60mm
Pulses per revolution - 168
Gear ratio - 3.5 : 1

Drive Logic

LMS Left Motor Set direction
LMT Left Motor Toggle

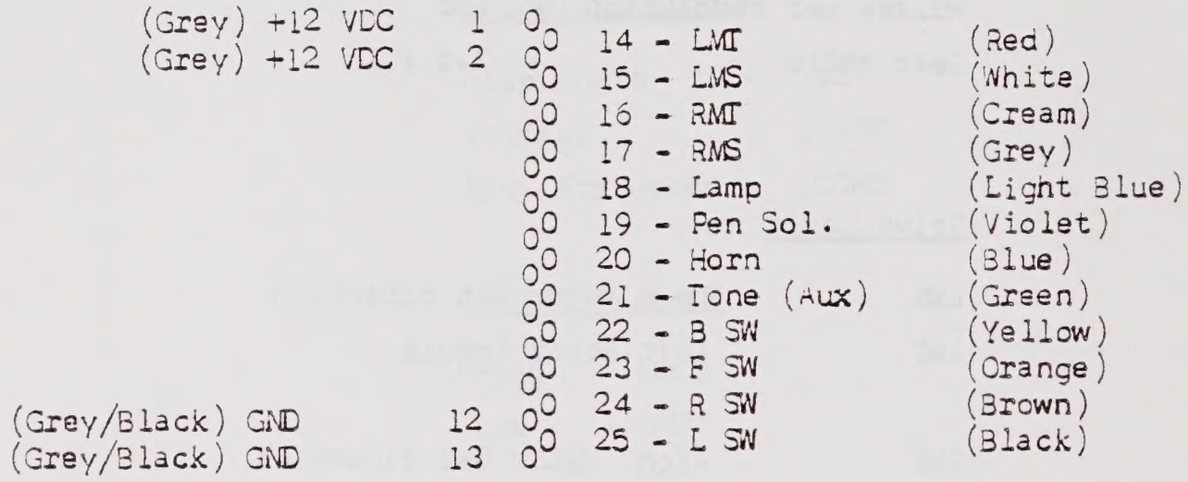
RMS Right Motor Set direction
RMT Right Motor Toggle

Set Direction High - clockwise
Low - anticlockwise

MOTION	LEFT MOTOR	RIGHT MOTOR
FORWARD	LOW	HIGH
REVERSE	HIGH	LOW
RIGHT TURN	LOW	LOW
LEFT TURN	HIGH	HIGH
RIGHT WHEEL (F)	LOW	OFF
LEFT WHEEL (F)	OFF	HIGH
RIGHT WHEEL (B)	HIGH	OFF
LEFT WHEEL (B)	OFF	LOW

Connecting Plug

25 Pin RS 232 Plug


(Grey) +12 VDC 1
(Grey) +12 VDC 2
14 - LMT (Red)
15 - LMS (White)
16 - RMT (Cream)
17 - RMS (Grey)
18 - Lamp (Light Blue)
19 - Pen Sol. (Violet)
20 - Horn (Blue)
21 - Tone (Aux) (Green)
22 - B SW (Yellow)
23 - F SW (Orange)
24 - R SW (Brown)
25 - L SW (Black)
(Grey/Black) GND 12
(Grey/Black) GND 13

Maintenance

It is anticipated that the Tasman Turtle will be relatively free of maintenance, however, as with all mechanical devices, certain adjustments may have to be made from time to time and dependent on the frequency and nature of use.

Adjustments are simple to make.

Sensor Switches

See Fig. 1

1. The Peripheral Sensor Band is held in a central position by the micro switch actuators.

2. To adjust the band loosen each of the four micro switch setting screws (a1, a2, a3, a4) and the four micro switch holding screws (b1, b2, b3, b4).

3. Adjust the band until the gap A1 = A2 = B1 = B2 = approx. 3mm.

4. Tighten the screws.

5. Check that each switch is actuated by the Peripheral Sensor Band.

6. Check that the band returns to its central position after each switch is actuated.

Note: The micro switch should not be adjusted out so far that it is always actuated, nor in so far that the Peripheral Sensor Band cannot actuate it.

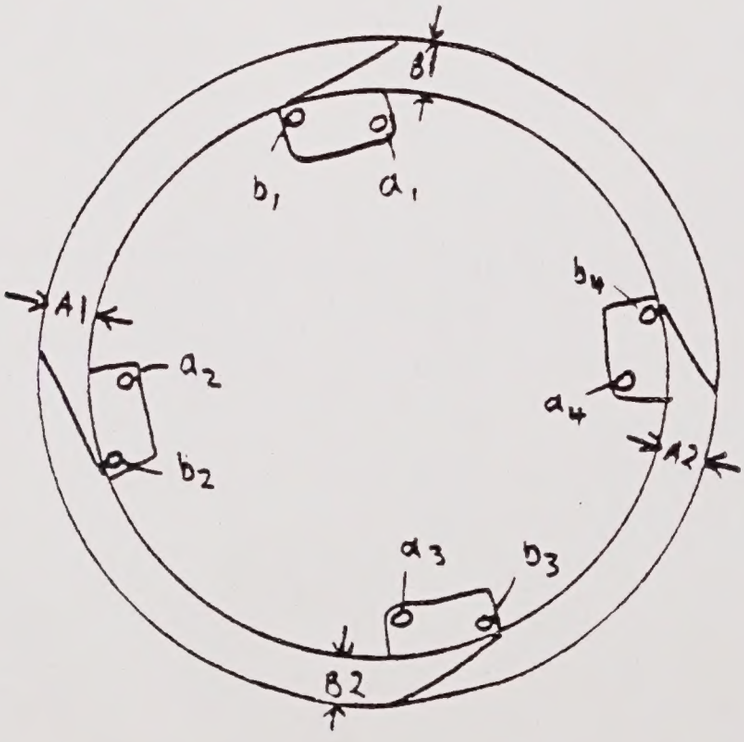


Fig. 1

Motor Drive

See Fig. 2

1. Adjust for good mesh (a) between G1 and G2 by loosening the angle bracket screw (b) and adjusting wheel (c) sideways.

2. Further adjustment is possible by loosening the motor front mounting screws and pushing G1 down on G2 gently.

3. Good mesh is obtained when any visible gap between G1 and G2 just disappears.

Note: Do not adjust the angle bracket so far that the wheel rubs on the base plate.

The motor drive mechanism will vibrate if held in the air while adjusting. Test all adjustments by running on a flat surface.

Too much pressure between G1 and G2 will cause the motor to miss steps.

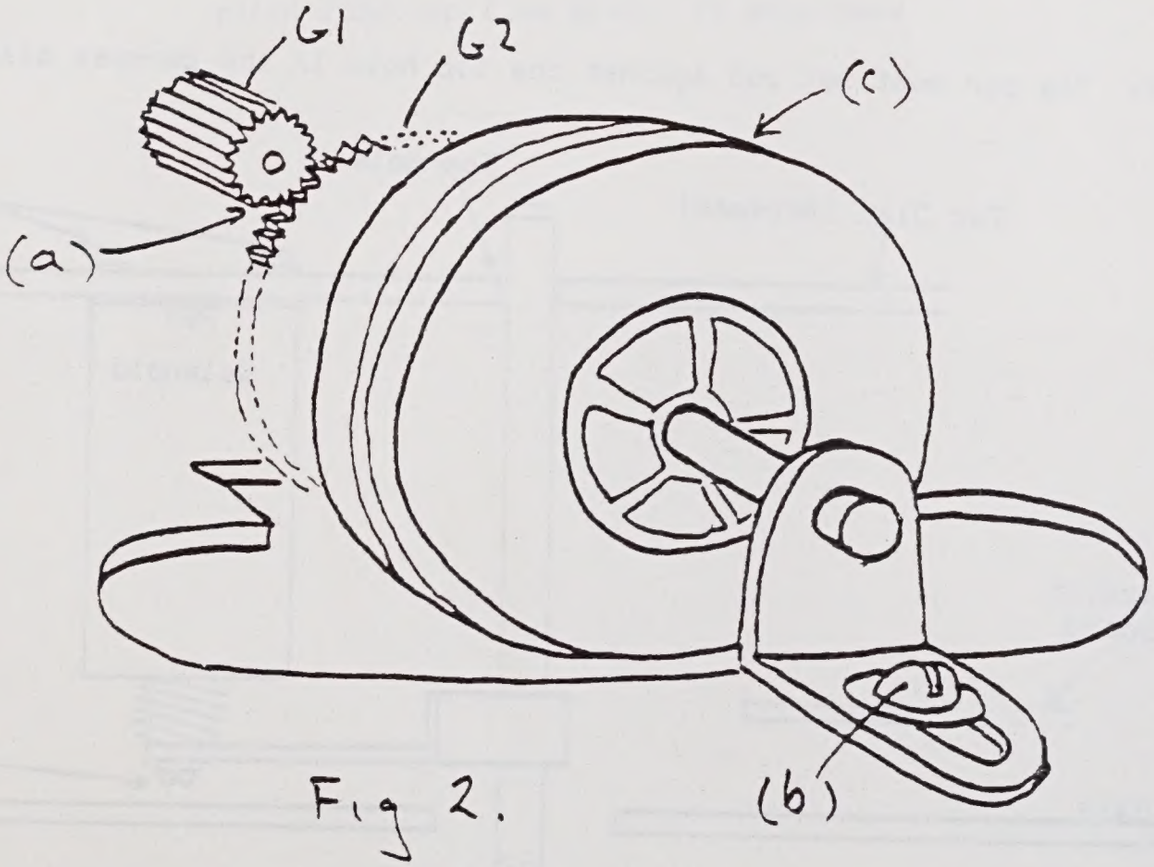


Fig 2.

Pen Adjustment

See Fig. 3

1. Centering of the pen is done by adjusting the pen solenoid mounting screws.

2. Remove dome by removing the four flange nuts on the long micro switch screws.

3. The dome will move off to the side, do not pull on the cable or plugs.

4. Adjust pen solenoid by trial and error till the best centering is obtained.

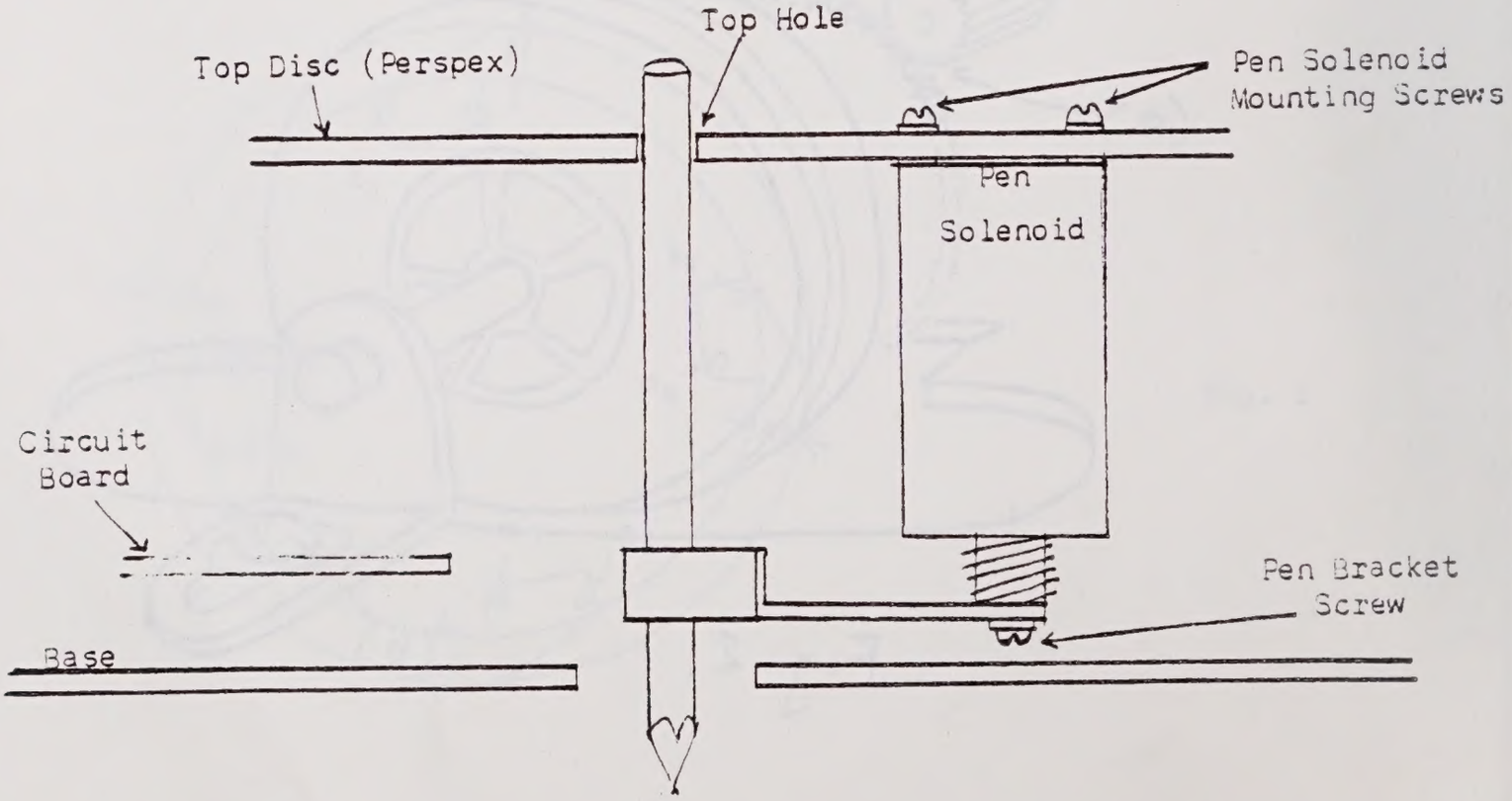
5. The best centering is obtained when the pen leaves a small dot or the smallest circle when the Turtle rotates 360°.

6. The turtle can be operated with the dome disconnected by simply placing the dome over the base, but not putting the four flange screws on.

7. If centering cannot be obtained satisfactorily then the Pen Bracket Screw may need to be adjusted. This should be done rarely.

8. Replace dome and flange nuts ensuring that the cable and plugs inside the turtle are satisfactory.

9. The pen must not rub against the top hole in the perspex disc.



Auxiliary Input

If a specific function is required but not provided, provision has been made for owner modifications.

1. Connect the horn-pitch input to the auxiliary input.

2. Connect the function device (solenoid, oscillator, light, Dc motor, etc.,) to the auxiliary output.

3. Test by applying 12VDC to the Horn/ Auxiliary input.

Note: That the two tone horn function is sacrificed to allow this modification, but the single pitch function (low pitch) is retained.

Terrapin - Apple Interface User's Manual

Deborah G. Tatar

Terrapin, Inc.
678 Massachusetts Ave.
Cambridge, MA 02139
(617)492-8816

Copyright © 1981 Terrapin, Inc.

99AD

General Introduction

These instructions should tell you everything you need to know about how to set-up, test and begin using your Terrapin-Apple Interface and Turtle. Each section begins with an overview of the essential instructions contained within that section and continues with details and explanations.

Parts List

Inside this box you will find two items in addition to these instructions:

1) your power supply -- a small heavy black box
2) a flat white cardboard box containing:
   a) a CCS Parallel Interface card with special ROM -- a flat circuit board packed in foam rubber
   b) the connecting ribbon -- a flat cord about 1 1/2 inches wide
   c) a booklet entitled Owner's Manual for a Parallel Interface Card
   d) two small black rectangular ROM (Read-Only Memory) chips with pointed silver leads

You only need parts 1, 2a and 2b to assemble and run the Turtle with the Terrapin - Apple Interface.

Items 2c and 2d should be stored some place safe in case you ever want to use your Parallel Interface card with a printer. The two chips (items 2d above) that you found packed in the foam rubber below your Interface card were originally on the card. Terrapin has replaced them in order to let you run a Turtle. However, it is possible to reinstall them, and the Owner's Manual... tells you how to manage the Interface card when those chips are in place.

To Set Up

Basically, all you have to do is plug the connecting ribbon in at both ends, that is, into the Interface card and into the power supply, then place the Interface card in slot 7 of the Apple and plug the power supply into a wall socket. The following instructions are to make sure you get the right ends plugged in the right way. Make especially sure to get the marked side of the ribbon lined up with the column of pins marked 1 on the connector of the Parallel Interface.

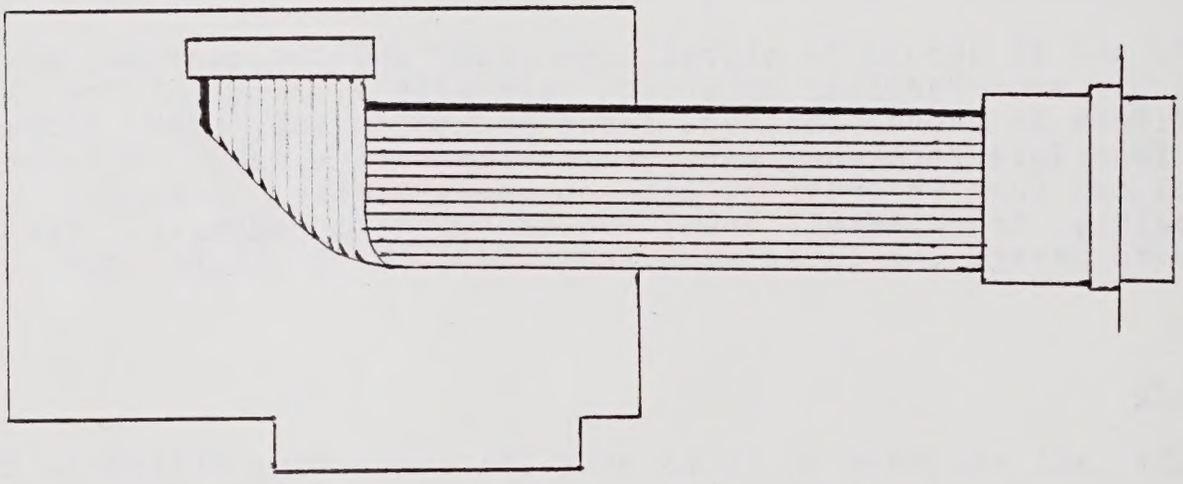
Before you start, turn the power off. Take the lid off your Apple. Touch the large gold box inside to discharge any static.

1) Find the flat ribbon and the Interface card. Turn the Interface so that the letters PI (for Parallel Interface) are upright and read correctly. In the top center of the card you will now find a silver-colored rectangle made of two rows of

small densely-packed pins. This is a connector. Underneath the first and last column of pins, you will find a small number printed on the board, labeling the pins 1 and 25. Locate number 1. Now find the small plastic end of the connecting ribbon. There will be either a colored strip on one side of the cable, or a triangular mark or some other tick that will distinguish between one side of the small end and the other. The side that is marked should be lined up with the column of pins marked 1 on the Interface and the ribbon should be (carefully!) plugged in. Be careful not to bend the pins -- make sure the sockets and pins are aligned. It requires a firm squeeze to slid into place. It is not completely plugged in if you can still see the metal connecting pins. There is usually a small snapping sound as it finally slips into place.

2) Orient the ribbon so that when you put the card into the computer, the ribbon will fit relatively smoothly into the slot in the back wall. To do this, fold the ribbon underneath itself and to the right. Do not crease the fold.

Illustration 1



Now plug the card into slot 7 in the computer. As you stand in front of the computer, this is the last slot on the right. The side of the card with the ribbon attached to it should face the right wall of the computer. There is an opening in the back wall of the computer behind slot 7. You should be able to slide the ribbon into it as you lower the card into place. Again, although one must exercise caution not to bend any components, it requires some firmness to get the card into the slot.

Now put the lid back on your Apple.

****
****

EUROPEAN MODEL Apples take the Interface card in slot 5 instead of slot 7. Therefore, whenever the slot number is invoked--in programming, as well as installation--owners of European model Apples must substitute 5 for 7.

****
****

3) Find the small (6.5" x 4.5" x 2.5") heavy black box that is your power supply. It has two ends. The front end is distinguished by a switch, and the back end by a small screw with a slot in the end. Put the power supply on top of the Apple so that the front end faces the keyboard. Now plug the loose end of the connecting ribbon, which should be hanging out of the slit behind slot number 7, into the back end of the power supply. The plug is not symmetrical; if it won't go in, turn it over and try again. Don't crease the cable.

4) Now plug the Turtle's cord into the front end of the power supply. Again, the plug is not symmetrical; if it won't go in, turn it over. You may want to secure the hooks on either side of the plug over the connection; however, you don't need to.

5) Finally, plug the power supply into a standard wall socket.

To Test the System

To test your Turtle and Interface card, you need to run each Turtle command and test all the touch sensors. The following programs show you how to do this.

If you plan to use your Turtle in teaching, this first experience can provide valuable data. Now how you approach the Turtle the first time. Your experience can be helpful when introducing the Turtle to the students.

Part 1: Checking the Turtle Commands

Flip the on-off switch on your power supply to the on position. It should glow red. If it doesn't, make sure you have plugged your power supply into a wall socket.

Turn on your Apple. Type in the following program: (1)

---

(1) To stop this program, press control-C when it asks for a command. Should you need to stop the turtle immediately, press the RESET key.

```
10 PR#7 : PRINT "*"
20 PR#0
30 INPUT "TURTLE COMMAND?"; A$
40 PR#7 : PRINT A$
50 GOTO 20
RUN
```

****
****

European Model Apples---substitute PR#5 for PR#7 in the above program.

****
****

This program asks for and executes Turtle commands. It prints out the words "TURTLE COMMAND?" When you type in your instructions and press RETURN, the Turtle should exhibit the desired behavior and the words "TURTLE COMMAND?" should appear once more.

The Turtle has thirteen built-in commands, falling into two categories: motion commands and auxiliary commands.

Motion commands:

Command	Means	Example	Produces
F <some#>	Forward <some#>	F 10	Turtle moves Forward 10 inches
B <some#>	Back <some#>	B 10	Turtle moves Back 10 inches
R <some#>	Right <some#>	R 90	Turtle turns 90 degrees Right
L <some#>	Left <some#>	L 90	Turtle turns 90 degrees Left
M <some#>	Right wheel forward <some#> left wheel still	M 90	Turtle pivots 90 degrees forward around left side
S <some#>	Left wheel forward <some#> Right wheel still	S 90	Turtle pivots 90 degrees around right side
Q <some#>	Right wheel back <some#> Left wheel still	Q 90	Turtle pivots 90 degrees back about left side
K <some#>	Left wheel back <some#> Right wheel still	K 90	Turtle pivots 90 degrees back around right side.

In other words, each letter command must be followed by a numerical argument, i.e. you must tell the Turtle how far to go forward or how much to rotate. For example, F 10 should make

your Turtle move forward 10 turtle units (roughly 10 inches) and R 90 should make it turn right 90 degrees.

None of the <some#>s may be greater than 255; larger numbers will produce generally unpredictable results.

Notice that rotating right or left is caused by both wheels turning in opposite directions. The four pivot commands, M, S, Q and K, involve only one wheel moving. Right and left rotations cause the center to stay still, while in pivoting one of the wheels remains stationary. All turning is caused by the relationship between the actions of the right and left motors.

If your Turtle does not behave as expected, there are several possible reasons. If it does nothing at all, you should double-check all the connections. Then try turning the screw on the power-supply all the way first in one direction, then in the other.

You will be told how to correct any curves produced by Forward or Back and any deficiencies in the amount of rotation produced by turn commands in the Adjustments section.

Hint: Try to use a smooth floor rather than a rug.

Auxiliary commands:

Command	Means
P <some#>	Pen-- Numbers > 0 put the pen down, 0 raises the pen
E <some#>	Eyes--Numbers > 0 make the eyes glow red, 0 turns them off
H <some#>	Horn--Numbers greater than 0 turn the horn on, 0 turns it off
T <some#>	Tone--Numbers > 0 switch the horn higher 0 restores its original tone
O	Off-- Raises the pen and turns off the eyes, horn and motor. Can take <some#>, but does not require one.

To get out of the program you typed at the beginning of this section, press control-C. If you wish to start it up again, type RUN.

Part 2 --Checking the Touch Sensors

Your Turtle has four touch sensors, left, right, back and front. To check them out, type in the following program:

```
1Ø PR#7 : PRINT "*"
2Ø PR#Ø
3Ø DEF FN TUCH(DUMMY) = 255-PEEK (49392)
4Ø X = FN TUCH(Ø)
5Ø PRINT X
6Ø GOTO 4Ø
RUN
```

When you press the left side of your Turtle, a series of 1s should appear on the screen. The right side should produce 2s, the front 4s, and the back 8s. These numbers are additive, so that pushing the right and the front will produce 6s, the left and back 9s and so on. Different combinations of one or two sensors being activated permit recognition of contact in any of 8 directions.

****
****

European model Apple users-- remember: use PR#5 instead of PR#7 for the PR#. Also, the PEEK number is not 49392 but 49360.

****
****

Again, to get out of this program, type control-C.

Adjustments

There are two ways to adjust your Turtle. The first is mechanical, and the second computational. Mechanical adjustments include the slotted screw on the back of your power supply which adjusts both motors equally, and the four screws in the small blocks (called "pots") on top of your Turtle which allow you to address the forward and back functions of each motor separately. Computational adjustments allow you to modify any Turtle motion through changing the delay increment in the Apple's memory.

The slotted screw on the back of your power supply adjusts both motors equally. Use this to adjust the amount of rotation per unit command. Test for the correct setting by turning your Turtle some constant amount such as 90 degrees. For example, type R 90 to the program listed on page 5 and then adjust the screw. Stop when the Turtle actually moves 90 degrees. (This change will also affect forward/backward movement slightly.)

Additionally, you can control the amount that each motor turns in either direction. Because the adjustments are separate for each motor, you can use them to correct any curvature in Forward or Back commands. (If the motors are not working equally, you will get curvature.) Because the adjustments are separate for Forward and Back motion, you must be sure to check both functions; it is possible to have both motors working equally to go Forward 10,

but not for Back 1Ø.

To make the adjustment, unscrew the dome of your Turtle. On top, in a line just front of the center, you will find four rectangular plastic blocks with small screws in them. They are the pots (short for "potentiometers") referred to above. Next to them on the printed-circuit board is written FD or BK. The two on the left (from the Turtle's point of view) affect the left motor and the two on the right, the right motor. The inside screws adjust the forward motion and the outside screws, the backward. Turning any screw towards the center of the Turtle causes the speed of that particular motor and direction to increase. For example, turning the rightmost screw (leftmost as you face it) toward the center causes the right motor to increase speed in the backwards direction. If your Turtle draws a curve when you command it forward or back, these adjustments allow you to make sure that your Turtle goes in a straight line, i.e. that the motors are working equally. You can also use these commands to ensure that the Turtle moves the same amount forward and backward per unit command.

Occasionally, the Turtle will move in a tight circle, or move intermittently when commanded to move forward or back. There is a slight possibility that a wheel has worked loose and needs to be tightened with the allen wrench which is supplied.

The computer does not actually measure how far the Turtle goes when it moves but rather how long the motors are on. Therefore, you can adjust the delay constant stored in the Apple's memory. This number represents the amount of time the computer allows the motor to be on for each unit of command. There are three constants which allow you to do this, FCONST, TCONST and ACONST affecting respectively, forward/backward motion, left/right commands, and K/M/S/Q commands. Initially, they are set to the highest constant possible. If this does not work out, you may lower them.

The way to do this in your programs is to follow the initialization of the Interface card (PR#7 : PRINT "*") with lines that express the desired modification. Just as the initialization must be repeated each time the Apple has been turned off or the RESET button pushed, so to must these modifications be added each time to the original program.

FCONST controls the amount of time the motor is on per unit command in forward and backward motion. It is normally set at 255. To change it, you must type in: POKE 1535, <some#-between-1-and-255>.

ACONST controls the amount of time the motor is on in right or left commands. It is normally set at 87. To address it type: POKE 1663, <some#-between-1-and-255>.

TCONST controls the amount of time the motor is on for the commands K, M, S and Q. It is normally set at 62 and may be changed by typing: POKE 1791, <some#-between-1-and-255>.

When trying to determine how much to change these constants, remember that the effect of an adjustment is not linearly proportional to the numerical change; the effect is proportional to the square of the ratio. For example, typing in POKE 1663, 174 will make turning angles not two times greater, but four times.

Format

Programming for your Turtle is just like any other programming in BASIC except for a few special considerations.

Initialization: every time you begin to use the Turtle or press the RESET button run a program beginning with the following lines:

```
10 PR#7 : PRINT "*"
20 PR#0
```

PR#7 tells the computer to send commands to slot #7 (where you inserted the Interface card). PRINT "*" initializes the control program on the Interface card. PR#0 tells it to make PRINT print on the tv-screen again.

Line 10 should be included in every program that is written for the Turtle. (1)

RESET: Pressing RESET has nearly the same effect as typing PR#0. You can press RESET at any time, even if the turtle is running. It will stop the turtle (and your program) immediately, and redirect printing to the screen. Note that pressing RESET while the turtle is running may disable DOS commands; type CALL 1002 if attempts to use DOS commands result in "SYNTAX ERROR." Turning the computer off and back on will fix most unusual conditions.

Touch-Sensors: Note that "TUCH", not "TOUCH" must be used in the touch-sensor example program because "to" is a restricted word in BASIC, and appears in the word "touch."

Touch-sensor numbers: Memory location 49392 (49360 on European Apples) is associated with the Turtle's touch sensor. PEEKing this location in BASIC returns a number from 0 to 255. If the turtle not touching anything, the number will be 255. For convenience, the TUCH function provided in the program on page 6 inverts these numbers so that PRINT FN TUCH(0) prints 0 when the

---

(1) European model Apples--here as always #5, not #7.

Turtle isn't touching anything. Here are the numbers returned by FN TUCH(Ø) when various sensors are touching something:

Left	1
Right	2
Front	4
Back	8

When two sensors are on, the number will be the sum of the two values for the respective sensors.

Other Programs: Turtle functions may be included in any program, even programs that also do entirely different things, by intializing the Interface card and giving a standard Turtle command of the form PR#7: Print "<somecommand> <some#>". A typical subroutine that could take an input produced by the rest of the program might look like this:

```
1ØØ PR#7 : PRINT "*"
11Ø PRINT TC$;TA : PR#Ø
12Ø RETURN
```

To use this as part of another program, assign the desired command to the string variable TC$, and the argument to the numeric variable TA. Then GOSUB 1ØØ.

Using Your Turtle

In general, you will probably want to use your Turtle in conjunction with the programs on pages 4 and 6 that allow you to give direct motion commands and receive direct feedback from the touch-sensors; however, it is easy to modify these programs and create new ones.

The program listed below is the one you used to test motion commands. It prints out the words "TURTLE COMMAND?", receives an input, executes that input, and requests a new input.

```
1Ø PR#7 : PRINT "*"
2Ø PR#Ø
3Ø INPUT "TURTLE COMMAND?" ; A$
4Ø PR#7 : PRINT A$
5Ø GOTO 2Ø
RUN
```

It accepts the commands discussed in the Testing section with the amounts outlined there:

Command	Means	Example	Produces
F <some#>	Forward <some#>	F 10	Turtle moves Forward 10 inches
B <some#>	Back <some#>	B 10	Turtle moves Back 10 inches
R <some#>	Right <some#>	R 90	Turtle turns 90 degrees Right
L <some#>	Left <some#>	L 90	Turtle turns 90 degrees Left
M <some#>	Right wheel forward <some#> left wheel still	M 90	Turtle pivots 90 degrees forward around left side
S <some#>	Left wheel forward <some#> Right wheel still	S 90	Turtle pivots 90 degrees around right side
Q <some#>	Right wheel back <some#> Left wheel still	Q 90	Turtle pivots 90 degrees back about left side
K <some#>	Left wheel back <some#> Right wheel still	K 90	Turtle pivots 90 degrees back around right side.
P <some#>	Pen-- Numbers > 0 put the pen down, 0 raises the pen		
E <some#>	Eyes--Numbers > 0 make the eyes glow red, 0 turns them off		
H <some#>	Horn--Numbers greater than 0 turn the horn on, 0 turns it off		
T <some#>	Tone--Numbers > 0 switch the horn higher 0 restores its original tone		
O <some#>	Off-- The letter "O" can be followed by a number, but does not require one.		

An example of something you might like to do for young children is to create an environment in which a single letter command produces an entire movement, i.e. instead of typing F 10 all the child would have to do is press F or any other key (perhaps with an arrow taped to it) to produce a forward movement of some predetermined amount. Such a program could look like this:

```
10 PR#7 : PRINT "*"
20 PR#0
30 PRINT "WHICH WAY?"
40 GET A$
50 C$ = "Z"
60 IF A$ = "F" THEN C$ = "F 10"
70 IF A$ = "B" THEN C$ = "B 10"
80 IF A$ = "R" THEN C$ = "R 30"
90 IF A$ = "L" THEN C$ = "L 30"
100 IF C$ = "Z" THEN 130
110 PR#7 : PRINT C$
120 GOTO 20
130 PRINT "WRONG LETTER, PLEASE TRY AGAIN"
```

```
140 GOTO 20
RUN
```

Another program causes the Turtle to move or turn away from whichever sensor is activated :

```
10 PR#7 : PRINT "*"
20 DEF FN TUCH(DUMMY) = 255 - PEEK (49392)
30 X = FN TUCH(0)
40 IF X = 4 OR X = 5 OR X = 6 THEN PRINT "B 10"
50 IF X = 8 OR X = 9 OR X = 10 THEN PRINT "F 10"
60 IF X = 1 THEN PRINT "R 90"
70 IF X = 2 THEN PRINT "L 90"
80 GOTO 30
```

Be creative! May you find your Turtle intellectually stimulating, and lots of fun!

Appendix

Using the Terrapin - Apple Interface from Assembly Language

The following information is a sample program and listing of machine locations relevant to the use of the Terrapin - Apple interface with 6502 machine-language programs.

A full listing of the assembly language program that runs the Turtle is available from Terrapin at no cost.

$47F Command: Contains the current command character in ASCII. the MSB is used as a flag for argument accumulation (the MSB should be low for all commands except "*" (INIT)).

$4FF Argument: Contains a binary integer that determines the delay for the motor commands.

$57F AUX: contains a bit pattern that preserves the state of the auxiliary functions (Eyes, Pen, Horn, Tone).

The contents of these three locations regulate the software delay for Turtle motor commands:

$5FF FCONST: Integer constant that determines delay for F(orward) and B(ackward) (normally $FF).

$67F ACONST: Controls delay for L(eft) and R(ight) commands (normally $3E).

$6FF TCONST: Delay for K, M, Q, and S commands (normally $57).

```
TITLE SAMPLE
;A sample assembly language program for controlling the Turtle.

FCONST=$5FF
ACONST=$67F
TCONST=$6FF

COMMND=$47F
ARGMNT=$4FF
AUX=$57F

SNDCMD=$C7ØØ        ;Turtle routine on Terrapin - Apple Interface Card.

;Call this routine to initialize the interface.

INIT:     LDA #$8Ø+'*
          STA COMMND
          JSR SNDCMD
          LDA #$8Ø+$ØD
          JSR SNDCMD          ;Send the command to the turtle.
                               (* for initialization).
;Store any desired values of the motion delay constants in FCONST, TCONST,
;and ACONST at this point.
          RTS

;Call this routine with the command name in A, the argument
;in X, and the bit pattern for auxiliary devices in Y.
;(See the Turtle owner's manual for bit masks.)
DOCMDA:   STY AUX
;...
;Enter at this point to keep the same auxiliary bits.
DOCMD:    AND #$7F            ;Turn off MSB, in case it's present.
          STA COMMND
          STX ARGMNT
          LDA #$8Ø+$ØD
          JMP SNDCMD          ;Do it.
END
```

Danny:
Leigh and I felt
you'd want to know
what Terrapin is doing
these days. Incidentally,
we really sent this
to this guy — Pat

6 January 1982

Terrapin, Inc.
678 Massachusetts Avenue
Room 205
Cambridge, MA 02139

Gentlemen:

SRI International is currently under contract to the U.S. Department of Defense to study possible applications of Artificial Intelligence and Robotics systems to military functions. Our interests include combat, combat support, logistics, and industrial/manufacturing functions, death and destruction.

Please send me descriptive information and cost data on any of your products which you feel might be applicable to these uses. It is important that the information include, if possible, your estimates of (1) the function(s) the system might perform, (2) system unit procurement cost (in quantities ranging from one to 10,000), (3) installation cost (if any) per unit, (4) annual cost of expendable supplies and spares per unit, (5) annual man-hours of labor required for operation and maintenance of one unit, and (6) system characteristics such as weight, size, power input, performance capacities, etc., with pictures.

The information you provide will be used only in SRI's analytical studies, and will not be considered an official offer by your company to supply the system(s) at the quoted prices.

Please call me at (415) 859-2592 if you have any questions.

Sincerely yours,

William Schubert

William Schubert
Senior Systems Analyst
Center for Defense Analysis

WS/gjn

SRI International
333 Ravenswood Ave. • Menlo Park, CA 94025 • (415) 326-6200 • Cable: SRI INTL MNP • TWX: 910-373-1246

Terrapin, Inc.

678 Massachusetts Avenue #205
Cambridge, Massachusetts 02139
617-492-8816

3:05am Tuesday, 19 January 1982

William Schubert
Center for Defense Analysis
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

Dear sir:

We must admit to some initial puzzlement at receipt of your communique of 6 January regarding possible applications of our robotics and artificial intelligence products to military functions.

However, always eager to contribute to the defense of our country from the ever-present threat, we put our best minds right to work on the problem and put together the enclosed report. We hope it will aid in your analysis.

Please note that the information we are providing is to be used only in your analytical studies, and is <u>not</u> to be considered an official offer by Terrapin to supply the systems at the quoted prices.

Please call us at (617) 492-8816 if you have any questions.

Sincerely yours,

Patrick G. Sobalvarro

Leigh L. Klotz
Senior Software Engineers
Terrapin, Inc.

PGS,LLK/bbng

Introduction

At Terrapin, we feel that our two main products, the Terrapin Turtle (TM), and the Terrapin Logo Language[1] for the Apple II, bring together the fields of robotics and AI to provide hours of entertainment for the whole family. We are sure that an enlightened application of our products can uniquely impact the electronic battlefield of the future.

The Terrapin Turtle (TM) is a small, versatile robot that can perform any number of complex tasks under computer control. A powerful AI programming language is necessary to realize the full potential of this advanced device.

The Terrapin Logo Language, developed at the Massachusetts Institute of Technology Artificial Intelligence Laboratory's Logo Group, is ideal for this application. The Logo language is a close relative of Lisp, the language used most widely in AI research. It fills the bill quite handily!

---

1. Logo trademark under license from Bolt, Beranek, and Newman, Inc.

Descriptive Information

1. Functions the system might perform.

While somewhat limited in range,[1] Turtles show great promise as all-terrain, high-accuracy system with excellent survivability in hostile environments. The Turtle's low observability, low vulnerability to ECM, and its multidirectional sensing capabilities make it an ideal reconaissance vehicle. Its ability to perform complex terminal maneuvering while pushing a heavy payload makes it a superb delivery vehicle as well.

Survivability

The Turtle enjoys very low observability, due to a minimal radar cross-section and an almost non-existent infra-red signature. In addition, its ground-hugging characteristics maximize terrain masking, resulting in lowered target acquisition by most classes of SSM and ASM threats. Its sophisticated sensors generate virtually no radar signature, so antiradiation missiles are useless against it. Furthermore, the Turtle is completely invulnerable to high-altitude SAM attacks.

With its powerful motors, the Turtle has very high initial acceleration. It climbs 45-degree grades with ease. Its maneuverability more than compensates for its somewhat low cruising speed. The Turtle can make a 180-degree turn in less space than any military vehicle currently in use by US forces, ground, air, or sea. With minor modifications, a Turtle could be constructed that could double its cruising speed for a terminal "dash" capability that would greatly enhance survivability in the endgame.

Because of the Turtle's compactness, with proper camouflage, weapons that depend on human vision are ineffective against it, as ground troops will find it hard to spot even at very close range.

Even if a suitable counter were found to all these properties of

---

1. Extended-range Turtle Mark I variants are currently in testing, however. See the section on Range.

the Turtle, it seems dubious that an enemy could afford to deploy counterweapons in sufficient numbers to nullify the possibility of defense saturation in the event of an all-out Turtle attack. The low system cost of the Terrapin Turtle (TM) make it an economic way to defeat defensive systems of increasing technological sophistication.

Range

The Mark I, Mod 0 Turtle has an effective range of some 3-4 meters, depending on its winding count. Range is most severely limited by the Turtle's C3,[1] but this limitation is trivial by comparison with the inherent advantages of wire-guidance. These advantages are discussed further below (see section Guidance).

Furthermore, our research department is currently engaged in the testing of a 100-mile C3 for the Turtle. The thrust of this research is towards the development of an Extended-Range Turtle II (TM). While this does result in a shorter tooth-to-tail ratio, we feel it could significantly enhance the battlefield capabilities of Turtle installations.

Guidance

The Terrapin Turtle (TM), like many missile systems in use today, is wire-guided. It has the wire-guided missile's robustness with respect to ECM, and, unlike beam-riding missiles, or most active-homing systems, it has no radar signature to invite enemy missiles to home in on it or its launch platform.

However, the Turtle does not suffer from that bugaboo of wire-guided missiles, i.e., the lack of a fire-and-forget capability. Often ground troops are reluctant to use wire-guided antitank weapons because of the need for line-of-sight contact with the target until interception is accomplished. The Turtle requires no such human guidance; once the computer controlling it

---

1. Computer Control Cable, often familiarly called the Turtle's "tail" in civilian applications.

has been programmed, the Turtle performs its mission without the need of human intervention. Ground troops are left free to scramble for cover.

But why stop there? Even more interesting scenarios are readily envisionable.

Because the Terrapin Turtle (TM) is computer-controlled, military data processing technicians can write arbitrarily baroque programs that will cause it to do pretty much unpredictable things. Even if an enemy had access to the programs that guided a Turtle Task Team (TM), it is quite likely that they would find them impossible to understand, especially if they were written in ADA. In addition, with judicious use of the Turtle's touch sensors, one could, theoretically, program a large group of turtles to simulate Brownian motion. The enemy would hardly attempt to predict the paths of some 10,000 turtles bumping into each other more or less randomly on their way to performing their mission. Furthermore, we believe that the spectacle would have a demoralizing effect on enemy ground troops.

Since the Terrapin Logo Language incorporates list structure, it is conceivable that someone will someday write an intelligent general problem-solving program in it. If this happened, truly intelligent guidance of Turtles would be possible, and since everyone knows that "smart" weapons are superior to "dumb" weapons, the Terrapin Turtle (TM) is superior to every other weapon developed to date.

Munitions

The Terrapin Turtle (TM) does not currently incorporate any munitions, but even civilian versions have a downward-defense capability. The Turtle can be programmed to attempt to run over enemy forces on recognizing them, and by raising and lowering its pen at about 10 cycles per second, puncture them to death.

Turtles can easily be programmed to push objects in a preferred direction. Given this capability, one can easily envision a Turtle discreetly nudging a hand grenade into an enemy camp, and then accelerating quickly away. With the development of ever smaller fission devices, it does not seem unlikely that the Turtle could be used for delivery of tactical nuclear weapons.

General

When controlled from a properly programmed computer, the Turtle can solve mazes, push and follow blocks, and draw complex geometric figures. In military terms. this means it can travel through mine fields and repel advancing troops. Large quantities of Turtles could conceivably be used to simulate battle plans. (Terrapin, Inc. does not supply software for these applications.)

The Terrapin Turtle performs the following functions:

Forward <centimeters>
Back <centimeters>
Left <degrees>[1]
Right <degrees>[1]
Penup
Pendown
Eyeson
Eyesoff
Hornoff
Hornhi
Hornlow

---

1. Convertible to mils for military applications.

2. System Unit Procurement Cost

Item	Unit Cost	Quantity
Turtle, Assembled	$599.95	1
Turtle, Assembled	$637.17	259
Turtle, Assembled	$599.95	10,000
Turtle, Kit	$399.95	1
Turtle, Kit	$424.76	259
Turtle, Kit	$399.95	10,000
Apple II Interface	$199.95	1
Apple II Interface	$212.35	259
Apple II Interface	$199.95	10,000

3. Installation Cost

The Terrapin Turtle is designed for installation at no cost by children and elementary school teachers. We feel that military installation cost should be under $10,000/unit.

4. Annual Cost of expendable supplies and spares per unit.

Ball-point Pen refills $0.59 (one spare included)

In the rugged terrain of the battlefield, under rigorous load conditions, it may be necessary to occasionally replace the TurtleTires (TM). Due to fluctuations in the world rubber market, quotation of exact prices is not possible.

5. Annual man-hours of labor required for operation and maintenance of one unit.

Operation costs for a Terrapin Turtle (TM) can essentially be reduced to cost of programmer time. Terrapin, Inc., offers demonstration software with every Apple II interface. If the functions provided by this software are satisfactory for military purposes, then no man-hours of programmer time need be tallied.

In any case, we feel that it may not be fair to consider operation costs of single units, because we purposely designed the Terrapin Turtle (TM) in such a manner that <u>every</u> Terrapin

Turtle (TM) can execute the same functions as every other Terrapin Turtle (TM). Thus there are no problems of software incompatibility, and a single team of programmers can, theoretically, operate any number of Turtles.

The Turtle's high reliability components and rugged construction minimize its life-cycle costs.[1] We feel that this should contribute to a high weapons-on-target-per-dollar potential. However, the Turtle's C3 will sometimes require maintenance, and, occasionally, rotation. Careful programming of the Turtle can reduce the need for rotation significantly. The challenge, simply stated, is to develop sophisticated, yet flexible, routing performance boundaries so as to maximize near-zero winding counts.

With the careful design of programs with provisions for evaluation of field-operation and recertification test results, together with failure/discrepancy analysis and corrective action, we feel that it should be possible to optimally extend the scheduled maintenance intervals and actually effect performance exceeding specifications. It can be expected that with a conscientious implementation of such programs very little time need be spent in C3 maintenance.

This estimate was, of course, computed taking only peacetime sortie rates into account. Under wartime conditions, we feel that up to a six-to-one increase in sorties might be experienced, and thus operationally ready (O.R.) rates can be expected to decrease accordingly.

---

1. It has long been the contention of our public relations department that the Turtle maintains itself. We feel that, if this is true, with a little work, Turtles might be made to maintain other devices, such as tanks, helicopters, and aircraft carriers. Thus we feel that a negative figure might even be appropriate for maintenance costs.

6. System Characteristics

The Terrapin Turtle (TM) weighs approximately 32 ounces. It measures approximately 8" in diameter, and is about 6" tall. Its power requirements are 15 volts at 2 amps. In normal trim, its power output is about 27.4 watts; however, an enhanced Turtle with "dash" capability would necessarily have higher power input and output specifications.

Use Infrared Communication for Remote Control

The Texas Instruments SN76832AN Infrared Remote-Control Receiver simplifies the tough job of receiving modulated infrared light.

Steve Ciarcia
POB 582
Glastonbury, CT 06033

Two dozen images of Howard Cosell danced before my eyes. Suddenly there were two dozen images of acid indigestion, and then, just as suddenly, Howard Cosell was back.

Undaunted, I stood my ground, which happened to be in the television department of a large store a few miles from home. I was wielding a small box covered with push-button switches, trying out a display of remote-controlled TV sets.

There has been a silent revolution going on in the TV remote-control business. Silent indeed. No longer do the control boxes emit ultrasonic impulses that drive all the dogs in the household into hysterics. Today's remote-controlled TV sets receive viewers' instructions on beams of infrared light.

Detecting an information-bearing infrared beam in an infrared-saturated environment is a signal-to-noise horror show.

The silent revolution has also made controlling the sets easier. Ultrasonic remote-control units, because of their complexity and cost, usually have had only two or three control channels, making channel changing a tedious, repetitive task and limiting the set functions that could be remotely commanded.

Infrared-light units are not only practical; they have become much more sophisticated, often making available thirty or more control channels for less than the cost of a pair of ultrasonic transducers.

To supply the demand for support circuitry to build remote-control systems, several semiconductor manufacturers (such as General Instrument, Hitachi, Signetics, and Texas Instruments) are producing integrated circuits that encode and decode the command information used in these TV remote controls. Most encoder chips are designed to accept a keyboard input and directly modulate an infrared light source. At the receiving end, the encoded data stream is decoded by a decoder chip and a few discrete components.

I decided to buy a pair of infrared encoder/decoder chips and build a

About the Author

Fifteen years ago, Steve Ciarcia (pronounced "see-AR-see-uh") gave up a promising career as a security guard for the Famous Writers' School to become an electrical engineer and computer consultant. He has experience in nuclear instrumentation, process control, digital design, product development, and marketing. In addition to writing for BYTE magazine, he has published several books, including Build Your Own Z80 Computer *(BYTE Books, 1981).*

When he's not working in his Circuit Cellar, he enjoys cooking such foods as Eggplant Siciliana and Shrimp Provençale and driving his DeTamaso Pantera sports car.

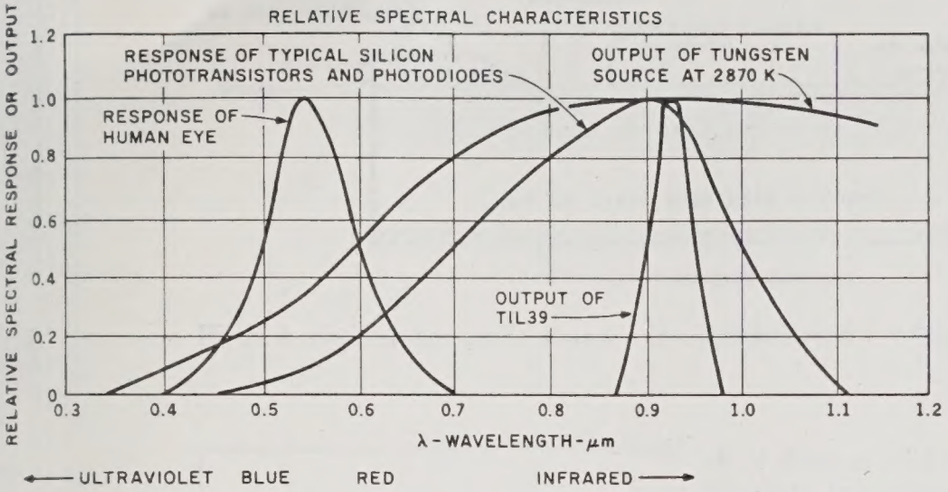


Figure 1: *Relative spectral characteristics of infrared light-emitting diodes and photodiodes.*

quick and dirty remote-control unit. It was going to be so easy.

Not so fast! There was a missing link: detecting the beam of infrared light so it could be decoded.

To Find a Missing Link

Was I going to let a little thing like a beam of light stop me? Of course not. I started investigating how to sense a beam of infrared light.

Once again I found myself working on a topic about which I could find very little published information of use in practical experimentation. I suspect there have been few magazine articles dealing with infrared-light communication because of the difficulty in detecting an information-bearing beam of infrared light in an infrared-saturated environment. It's a signal-to-noise horror show! The sensitivities and dynamic ranges required are beyond simple amplifier-design techniques. My goal of building an inexpensive remote-control scheme using TV-set encoder/decoder chips therefore had to wait until I first designed a reliable infrared communication receiver.

This article contains both a discussion of optical communication and a construction project of a useful infrared-light transmitter/receiver interface for use with a personal computer. Along the way I attempt to answer some basic questions including the following: In what kind of applications can light transmission be useful? Why use infrared rather than visible light? What is the best choice of optoelectronic components for each application?

Why Use Light?

Light is used in communication for two major reasons: the medium's immunity to certain forms of interference and the relative ease of providing security for the communications link. Electrically noisy motors and other equipment generate electromagnetic interference (EMI) that can play havoc with radio-wave transmissions, and ambient noise can disrupt ultrasonic communication. These problems do not bother light beams, and a protected line-of-sight beam path or waveguide provides fair security against unauthorized interception.

Light can also be used in providing physical security for premises; a prowler might step over a tripwire, but he wouldn't know to avoid an invisible shaft of infrared light aimed at a detector in an alarm system.

What Is Infrared Light?

The segment of the electromagnetic spectrum that we perceive optically as visible light is narrow. On both sides of this band of visible light are regions of radiation that we can't see but which otherwise exhibit similar optical properties. Radiation of wavelengths shorter than we can see is called ultraviolet (UV) light, while wavelengths longer than we can see form infrared (IR) light. Any warm object radiates some amount of infrared radiation.

(Some may quibble with my use of the term "light" to discuss radiation that cannot be seen. I feel that any radiation that can be manipulated optically [by lenses and the like] should be called light, and that's how I am using the term in this article.)

The spectral graph of figure 1 shows that the visible band has wavelengths between about 400 and 700 nm (nanometers). Within the range of 400 to 700 nm, the different frequencies are perceived as different colors. For example, a light beam of 550-nm wavelength is perceived as green. What we perceive as white light contains all the visible frequencies.

Transmitting information on a beam of light is done much the same way as on a radio wave. The light must be amplitude-modulated at some carrier frequency, say 40 kHz. This allows the receiver to differentiate between the light coming from the transmitter and unmodulated ambient light. The data to be transmitted can be modulated onto the carrier in a number of ways, including amplitude and frequency modulation, pulse-width modulation (PWM), and pulse-code modulation (PCM).

For my application of a simple optical remote-control system, the less complicated PCM technique seemed best. This merely consists of turning the 40-kHz-modulated light on and off. At the receiving end, the presence of light is interpreted as a logic 1 and the absence of light as a logic 0.

Why use so high a carrier frequency? We have to use a frequency high enough that the communication is not susceptible to interference. The operating environment of our infrared system may contain such sources of interference as fluorescent lights, which flash at 120 Hz, or television sets, whose screens emit light with interference patterns at over 15 kHz. For open-air optical communication, frequencies at or above 40 kHz are preferred.

Any electrically excited light source can be amplitude modulated, but not

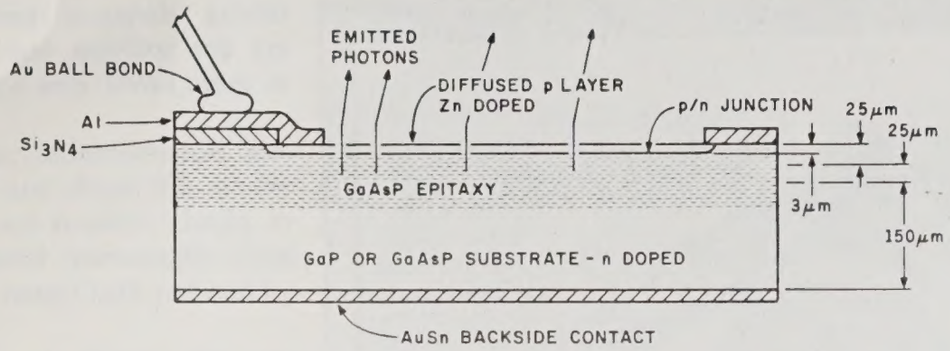


Figure 2: *Cross section of the structure of a standard visible red light-emitting diode.*

Blue LEDs at Last

Longtime readers of BYTE may recall Steve's article from several years ago on a self-refreshing LED graphics display (see reference 2). In it, he described how single-color graphic images can be formed on display panels containing arrays of many red light-emitting diodes and suggested that multicolor images could be displayed on a panel that contained arrays of triads of different-color LEDs. But his suggestion was ahead of technology. At that time, no practical method was known for making an LED that could emit blue light.

Now the blue-light barrier has been broken, and in a way eminently suitable to flat-panel graphics displays. In October 1981, the Sanyo Electric Company of Japan unveiled an LED lamp component that contains two separate LED chips, one of which emits blue light at a wavelength of 480 nanometers (see reference 3). The blue-emitting chip is formed from two liquid-phase-epitaxial layers of silicon carbide grown on a silicon-carbide wafer. The other chip, made of gallium phosphide with four epitaxial layers, can emit both red light at 700 nanometers and green light at 565 nanometers. A single component that can produce all three additive primary colors makes possible a full-color, flat-panel LED graphics display.

The voltage potential dropped by the blue LED chip is about 3.5 V, producing 2 mcd (millicandelas) of light, while the two-color chip drops 2 V for 3 mcd of green light and 1.9 V for 3 mcd of red light. Thus we see that real components reflect the theoretical predictions of greater efficiency at longer wavelengths. . . . **RSS**

all of them at 40 kHz. Modulating a 100-watt incandescent light bulb at such a high frequency is out of the question: the thermal time constant of the filament is much too long. The only light sources capable of switching at such a frequency are electroluminescent devices, of which the light-emitting diode (LED) is the least expensive and most familiar.

In a light-emitting diode, shown in figure 2 on page 42, light is generated when a forward-bias current is applied. This causes electrons to be injected into the n-type (negative-doped, electron-rich) semiconductor material and holes (shortages of electrons) to be injected into the p-type (positive-doped, electron-poor) material.

When the injected electrons and holes recombine with the majority carriers at the p/n junction, energy is released in the form of photons. The pattern of radiation emission can be controlled somewhat by reflective surfaces within the mounting structure and by plastic lenses. Generally, a spherical dome lens with a narrow beam width is best for communication.

The color of the light emitted depends upon what semiconductor materials are used in the p/n junction and how they are doped (seeded with selected impurities): the amounts of energy released in the electron/hole recombinations of different materials vary, and the wavelength (and therefore the color) of light varies in a direct relation to the amount of energy contained in its photons.

Most of the semiconductor materials used in LEDs are compounds of gallium: gallium arsenide phosphide (GaAsP, which emits red, orange, or yellow light), gallium phosphide (GaP, green emitting), gallium aluminum arsenide (GaAlAs, red emitting), and gallium arsenide (GaAs, which emits infrared photons at about 900 to 1000 nm). The efficiency of an LED depends upon wavelength. The longer the wavelength, the higher the efficiency (see the text box "Blue LEDs at Last").

Operating at longer wavelengths, infrared LEDs are more efficient than visible-light red or green LEDs; IR LEDs are therefore preferred for line-of-sight beam breaking or communication. For a given power input, an IR LED produces a brighter light than a green LED. Many IR LEDs have radiant-power outputs of more than 10 mW (milliwatts)—for instance, the TIL39 component I have been experimenting with has an 11-mW output at 940 nm.

Efficiency is important, because an LED is almost a point source of light. The illumination it casts on a surface is proportional to its brightness and inversely proportional to the square of the distance to the surface. If we want our open-air communication link to operate over distances as great as 10 meters, our LED light source must be very bright to produce an acceptable signal-to-noise (S/N) ratio using an ordinary-sized receiving photosensor measuring perhaps 0.1 cm^2 (square centimeter).

Other incentives for using IR light are reduced sensitivity to ambient visible light, greater ability to pierce through fog or smoke, and better reflection off most surfaces for a greater chance that the receiver will be able to see the light source.

Choosing the Light Detector

Choosing the proper light detector is as important as selecting the light emitter. Selectivity, response, and inherent noise are important considerations.

There are many materials which function as photoconductors. The simplest are bulk materials such as cadmium selenide (CdSe), lead sulfide (PbS), and cadmium sulfide (CdS).

Generally these materials exhibit poor temperature stability and are sensitive over a very broad range of wavelengths.

The familiar semiconductors germanium (Ge) and silicon (Si), on the other hand, are sensitive chiefly to the near-infrared (wavelengths close to the visible range) light radiated by IR LEDs.

Two types of devices that use silicon and germanium are photodiodes and phototransistors. Phototransistors are constructed in much the same manner as commonplace planar-diffused transistors. The base area is usually made large to provide an area into which incident light can penetrate and generate electron/hole pairs. Phototransistors are subject to the variation in performance typical of all transistors.

Many of the bad traits of phototransistors are eliminated in *PIN photodiodes*. The term PIN is an acronym meaning that the components are made from three layers of different types of semiconductor material: p-type, "intrinsic," and n-type. A PIN photodiode is one in which two heavily doped p and n regions are separated by a lightly doped region (which exhibits mostly properties intrinsic to the substance). Its large depletion region (interface region between the p-type and intrinsic layers) provides the PIN photodiode with much faster speed, lower noise, and greater efficiency at longer wavelengths.

Photo 1 shows a TIL413 infrared-sensitive PIN photodiode (right) and a TIL39 infrared light-emitting diode (left).

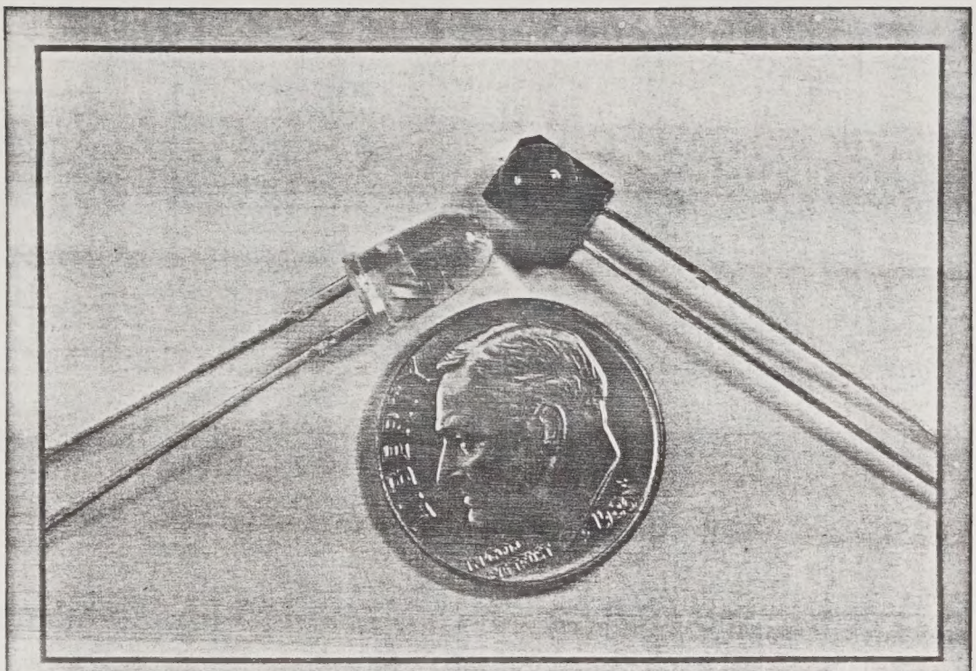


Photo 1: *The TIL413 infrared-sensitive PIN photodiode (right) and the TIL39 infrared light-emitting diode (left).*

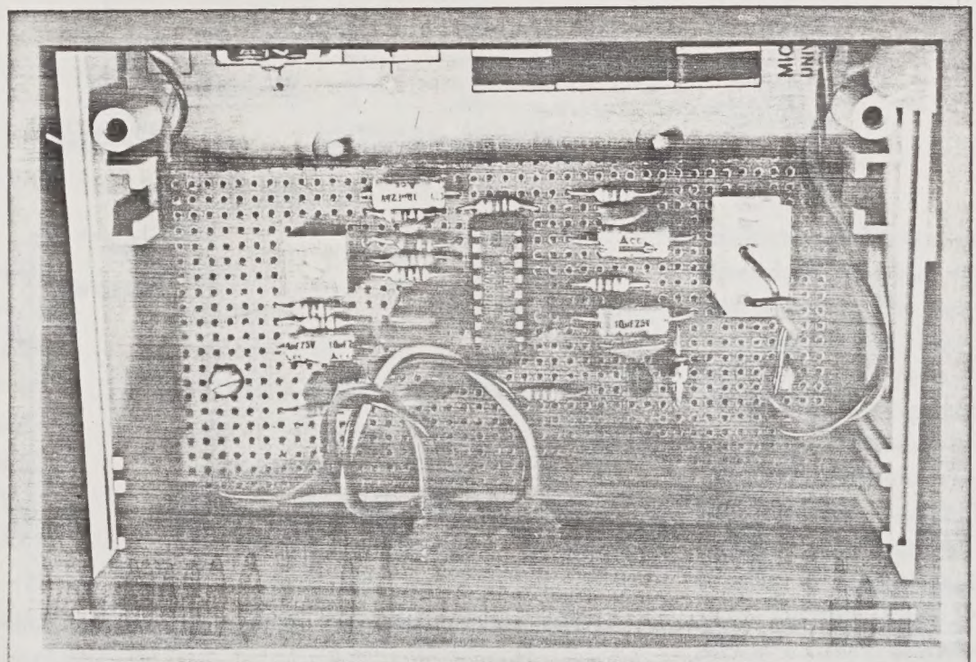


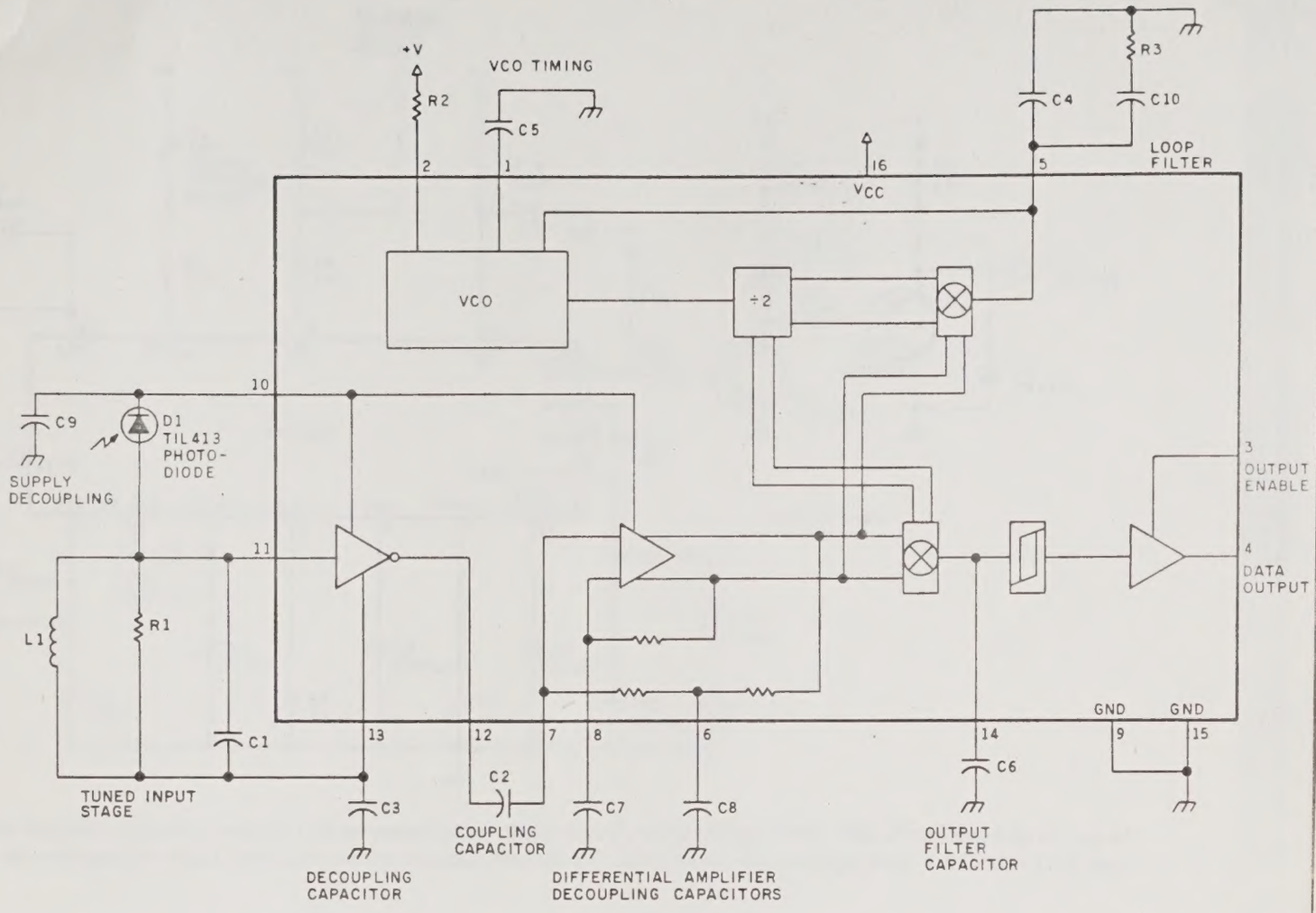
Photo 2: Prototype of the infrared-light remote-control or communication receiver of figure 4.

Ready-Made IR Detector

Designing a reliable infrared-light detector/receiver is no simple task; it has been the major obstacle in designing any infrared communication system. The engineer must coax his receiver into extracting the transmitted data from a dismaying amount of background noise, and he must take care that his design will withstand the impairments to theoretical performance caused by deviations from ideal component values and manufacturing techniques. But there is nothing conceptually complicated in the receiver, just a photodiode and a series of amplifiers and filters.

I haven't presented an infrared communication system before now because I couldn't design one that I felt readers could successfully duplicate. But recently I discovered that an integrated circuit has been developed to do all the hard work. Texas Instruments recognized the need and designed a chip which eliminates all the frustrations in building the IR receiver.

The new component is the SN76832AN Infrared Remote-Control Receiver. This chip replaces a combination of several integrated cir-



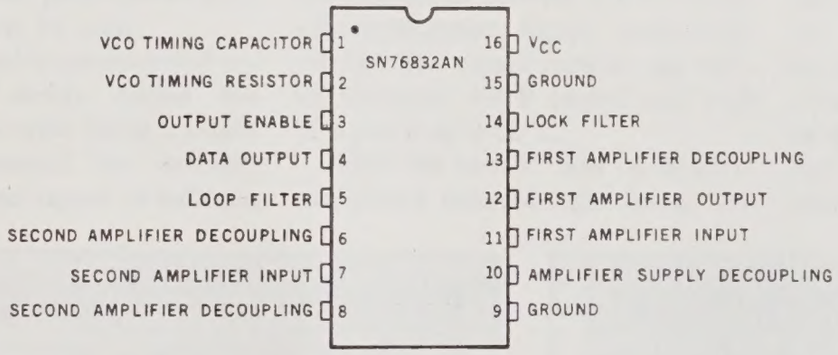


Figure 3: *Block diagram of the internal structure of the Texas Instruments SN76832AN Infrared Remote-Control Receiver.*

cuits and discrete components. It connects directly to a PIN photodiode and is designed to receive and detect digitally encoded information modulated on a carrier (typically 40 kHz). It has an open-collector gated output suitable for direct operation with a microprocessor. For the benefit of readers of BYTE, the Micromint will be distributing a kit that includes the SN76832AN (see page 49).

A block diagram of the SN76832AN receiver chip is shown in figure 3 . Inside the receiver chip, the signal from the photodiode is connected to an amplifier with an input impedance of 220 kilohms and a typical gain of 1.6 million volts/amp at 40 kHz. The output of this first amplifier stage is fed to a differential amplifier coupled to the demodulator section.

The receiver chip demodulates the signal by comparing its amplitude and phase with that of reference signals produced by a voltage-controlled oscillator (VCO) and a frequency divider. The bandwidth and capture frequency are controlled by external passive components. The de modulated output signal is filtered and gated by the output-enable con trol signal. (A high level on the output-enable line causes the demod ulated signal to appear on the outpu line. If the demodulated signal is to b constantly present, the output-enabl line should be tied high.)

Figure 4 on page 45 is a schemati diagram of the completed circuit for very sensitive IR receiver which oper ates at a carrier frequency of 40 kHz A prototype is shown in photo 2.

In the receiver circuit of figure 4 potentiometer R1 sets the frequenc of the VCO, which is twice that of th capture frequency (the center fre quency of the incoming modulate carrier signal). With the externa oscillator-control components shown this circuit can detect carrier frequen cies from 20 kHz to 70 kHz. The res of the components, however, are op timized for 40-kHz operation. Th photodiode I suggest is a type TIL41 because it has a spherical lens tha allows it to "see" over a wider angl

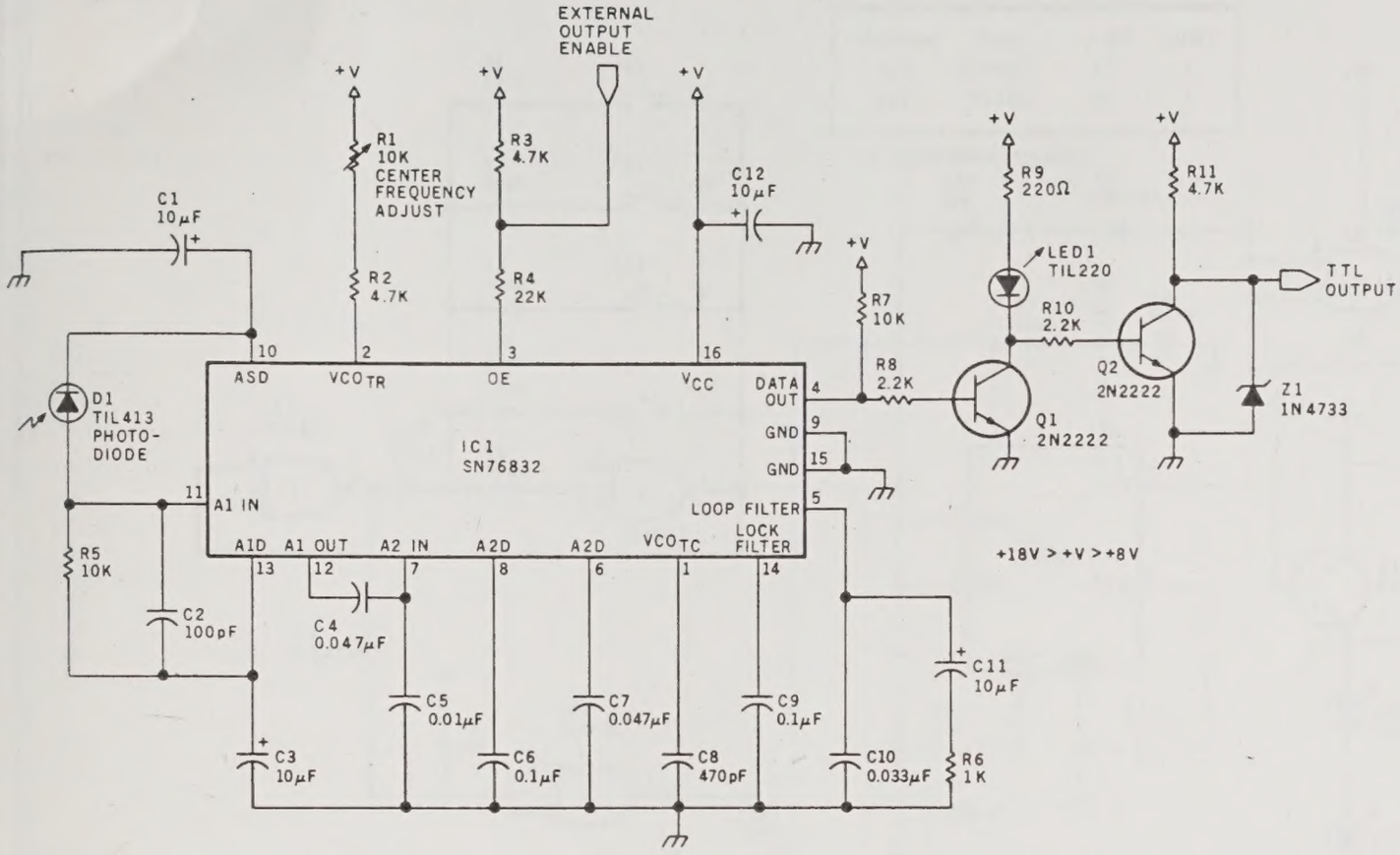


Figure 4: *Schematic diagram of a remote-control or communication circuit based on the SN76832AN. This circuit is set up for use at 40 kHz, although the voltage-controlled oscillator can be adjusted from 20 kHz to 70 kHz. Any voltage from +8 V to +18 V may power the circuit.*

thus enhancing the effective sensitivity. In a pinch other photodiodes such as the TIL100 may be used.

The data output is permanently enabled in my design unless the external-output-enable line is brought to ground potential by external means. The output signal is buffered and level-shifted through transistors Q1 and Q2 to provide a TTL- (transistor-transistor logic) compatible output. The circuit runs at any voltage between +8 V (volts) and +18 V. I run it at +12 V.

When the circuit "sees" a 40-kHz-modulated infrared light beam, the output goes to a normally high state, and the indicator light (LED1) comes on. The output then changes state in accordance with the demodulated data, exactly duplicating the sequence of the logic levels of the input signal that was fed into the infrared transmitter.

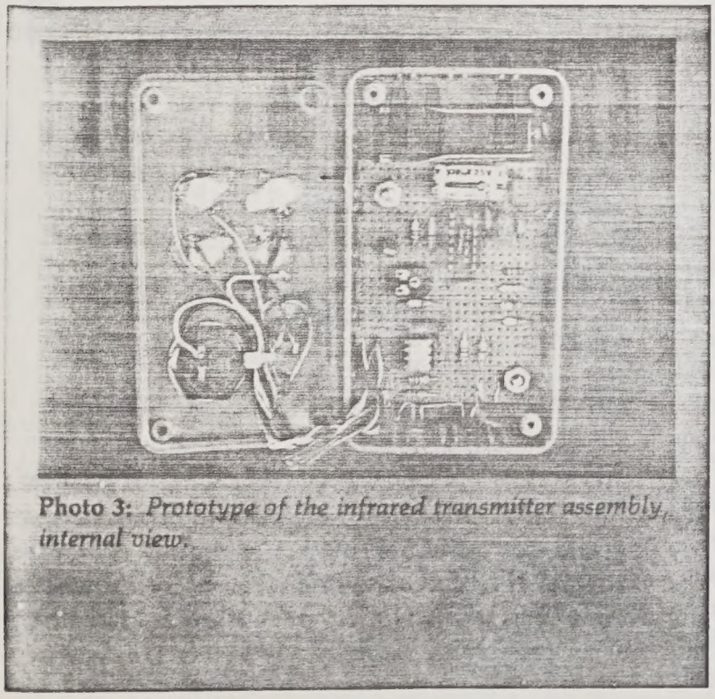


Photo 3: *Prototype of the infrared transmitter assembly, internal view.*

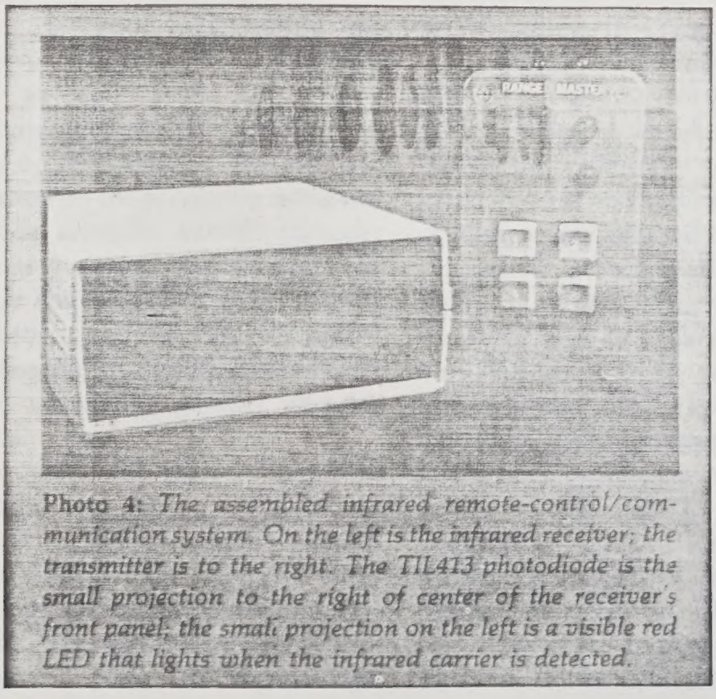


Photo 4: *The assembled infrared remote-control/communication system. On the left is the infrared receiver; the transmitter is to the right. The TIL413 photodiode is the small projection to the right of center of the receiver's front panel; the small projection on the left is a visible red LED that lights when the infrared carrier is detected.*

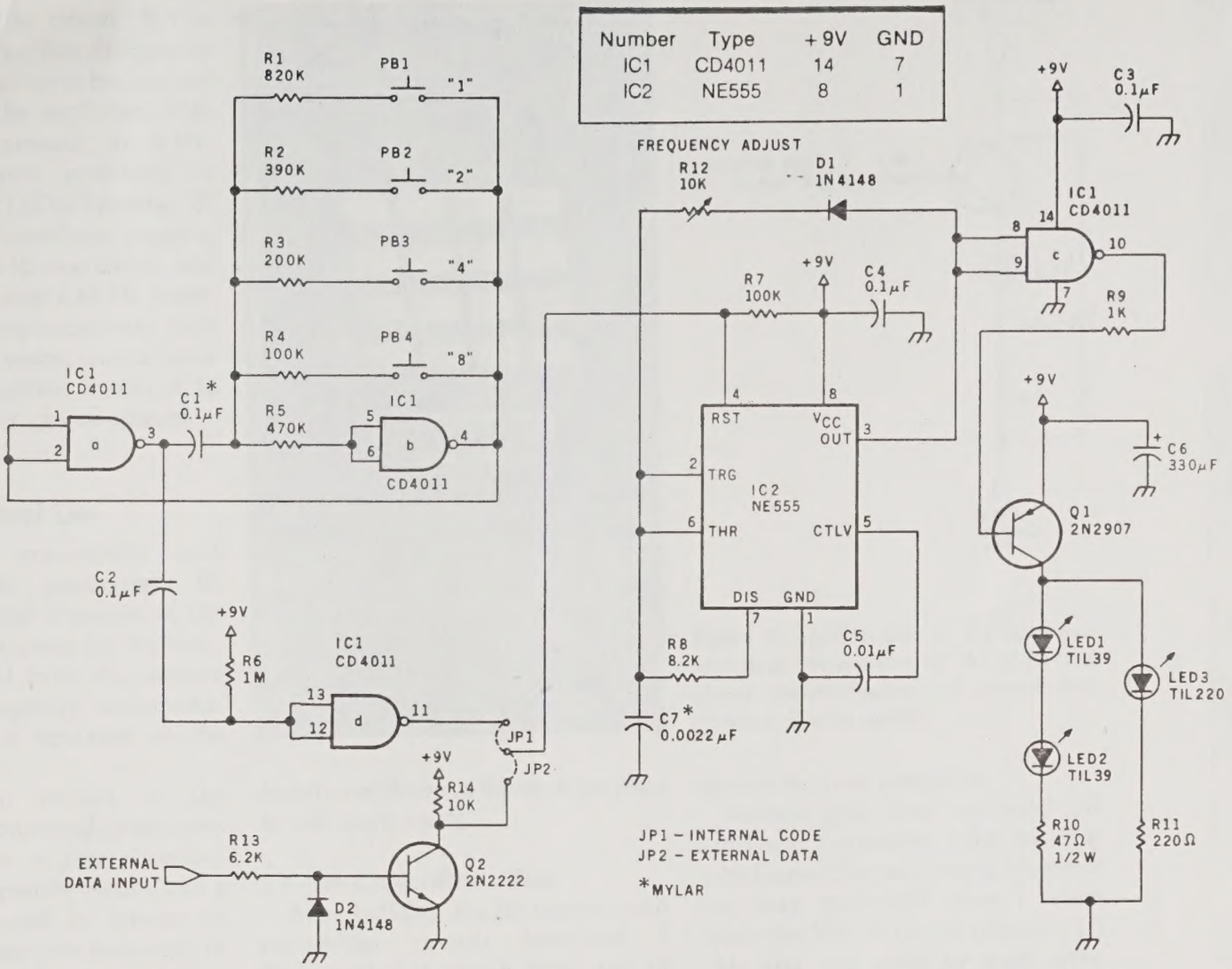


Figure 5: *Schematic diagram of a handheld infrared remote-control transmitter. Four pushbutton switches actuate a NAND gate wired as an oscillator; different frequencies may be selected for testing or in communicating simple command functions. Two high-power infrared and one visible indicator light-emitting diodes are modulated at a carrier frequency of 40 kHz; control signals are sent as a simple on/off modulation of the 40-kHz-modulated light beam.*

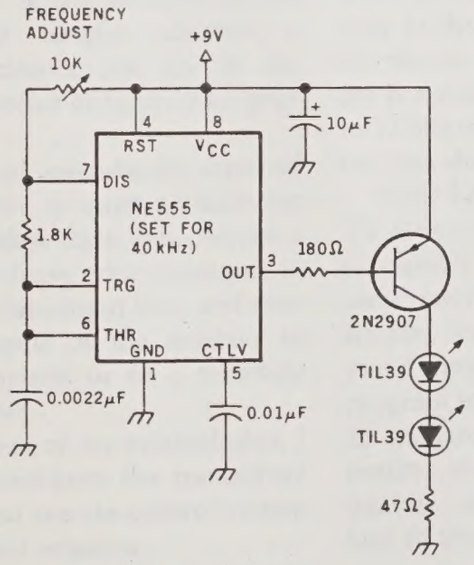


Figure 6: *A simple circuit that provides only a 40-kHz carrier modulation for an infrared beam. This may be used for simple beam-breaking intrusion-detection schemes.*

A Matching Transmitter

After building and testing the receiver section, I argued with myself about an appropriate transmitter. The "hardware" side of me wanted to follow up on the inspiration that got me started on the project by using one of those complex encoder/decoder chip sets previously mentioned. The encoder chip would be built into the transmitter, and the decoder would be connected to the infrared-receiver circuit, providing parallel decoded outputs.

The "software" side of me (I do have one!) argued that anything these chips could do in encoding and decoding could be done by any fairly competent programmer in a few lines of code. The real challenge of the project was creating the infrared-light communication link, not encoding and decoding the signal.

In the end, I decided that my inspiring vision was not worth the trouble, and I designed a relatively simple transmitter essentially consisting of a gated 40-kHz oscillator driving a pair of high-power infrared LEDs. As you can see in the schematic diagram of figure 5, jumper connections on the oscillator can be changed to allow modulation from an external signal source or to allow the communication link to be tested by your pushing one of the four pushbutton switches connected to the CD4011 NAND-gate component (IC1). The prototype transmitter circuitry is shown in photo 3 on page 45. The assembled transmitter and receiver are shown in photo 4 on the same page.

In the transmitter circuit, IC1 is configured as a low-frequency oscillator. The four switches control the frequency of the oscillator. With the "1" button pressed, an 8-Hz, 50-percent-duty-cycle waveform is directed to the IR LEDs. Pressing "2" produces a 16-Hz waveform, pressing "4" produces a 32-Hz waveform, and pressing "8" produces a 64-Hz waveform (the exact frequencies may vary somewhat). By pressing two or more of the buttons together, a total of 16 distinct frequencies can be created.

System Testing and Use

To test the transmitter and receiver, aim the transmitter's IR LEDs in the general direction of the receiver unit and press the buttons. The output signal from the receiver should be the frequency corresponding to the switch operated on the transmitter.

The single-bit output of the receiver can be connected to any convenient input line on your computer (such as a game-paddle input), and a simple program can be written to count and determine the frequency of the input signal. While much higher frequencies can be transmitted by this system, I chose these low, pulsed frequencies so that even a BASIC program could count the frequency. Nevertheless, if the frequencies are still too fast, simply substitute a higher capacitance for C1 in the transmitter circuit of figure 5 on page 46.

The external modulation input of the transmitter is quite suitable for use in a wireless data link. Attach a serial output from your computer to the external data-input line, and connect the output of the receiver to another computer or to a remotely located printer.

In fulfillment of my original plan, I decided to configure the transmitter in my infrared remote-control system as a hand-held actuator.

If you simply want an infrared beam source for use as an intrusion detector in a security system, you don't have to build the entire data-transmission circuit; the simple 40-kHz oscillator in figure 6 on page 46 will work nicely.

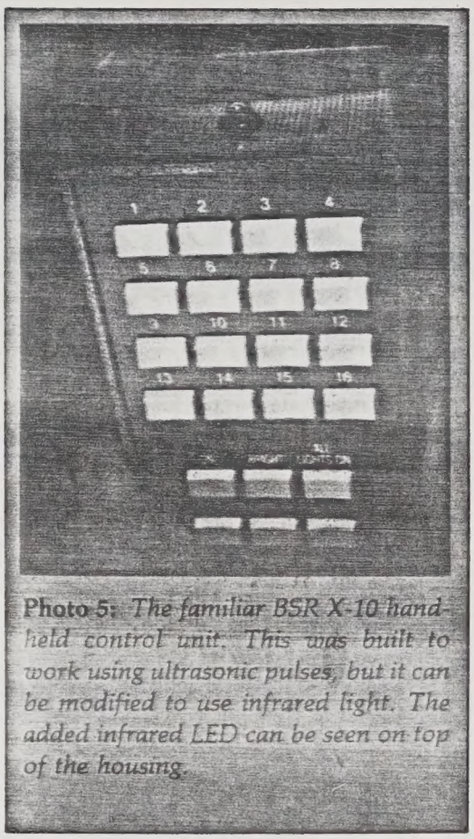


Photo 5: *The familiar BSR X-10 hand-held control unit. This was built to work using ultrasonic pulses, but it can be modified to use infrared light. The added infrared LED can be seen on top of the housing.*

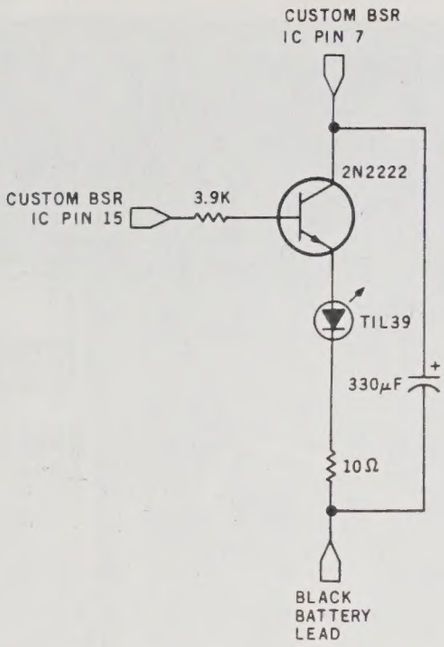


Figure 7: *Modification of the hand-held ultrasonic transmitter of the BSR X-10 Home Control System that converts it to infrared-light operation.*

TV-Set Controllers, Too

After building the IR receiver and transmitter circuits described, I discovered that signals from any of the commercial TV-set remote-control transmitters could be received just as easily as those from the transmitting device I had designed. Most set controls operate with carriers in the 38-kHz to 41-kHz range. Of course, the data output that you get is a coded bit stream, generally 5 to 12 bits repeating every half second, but this should be no obstacle.

While I haven't analyzed the coded TV-set-control data, they should be susceptible to differentiation by the same methods that worked for my simple frequency input. Perhaps you'd have to use a machine-language program to catch the fast data (about 25 milliseconds per word), but the results would be a professional-quality, very versatile remote interface to your computer.

BSR Goes Infrared

The remote-control transmitters for television sets are not the only means to an elegant remote-control system for your computer.

Perhaps you have on hand the hand-held transmitter from the BSR X-10 Home Control System. (Some of you may remember when I wrote about the BSR X-10; see reference 1.) This unit was made to work using ultrasonic sound, but it can be easily adapted for IR transmission to work with the receiver circuit in this article. Photo 5 shows the adapted BSR X-10 transmitter.

The modification is outlined in figure 7 and shown in photo 6 on page 48. Note that the ground connection should be the black lead directly connecting to the battery. The other two circuit connections are to pins 7 and 14 on the integrated circuit as illustrated. The BSR transmitter already operates using a carrier frequency of 40 kHz. The modification is simply to add an emitter-follower IR LED driver to the output section. The existing ultrasonic transducer can be cut out or used concurrently with the optical transmitter.

The message transmitted by the X-10 hand-held unit is approximately 100 milliseconds (ms) long and is composed of 13 eight-millisecond segments, each of which consists of a burst of 40-kHz signal. The reception

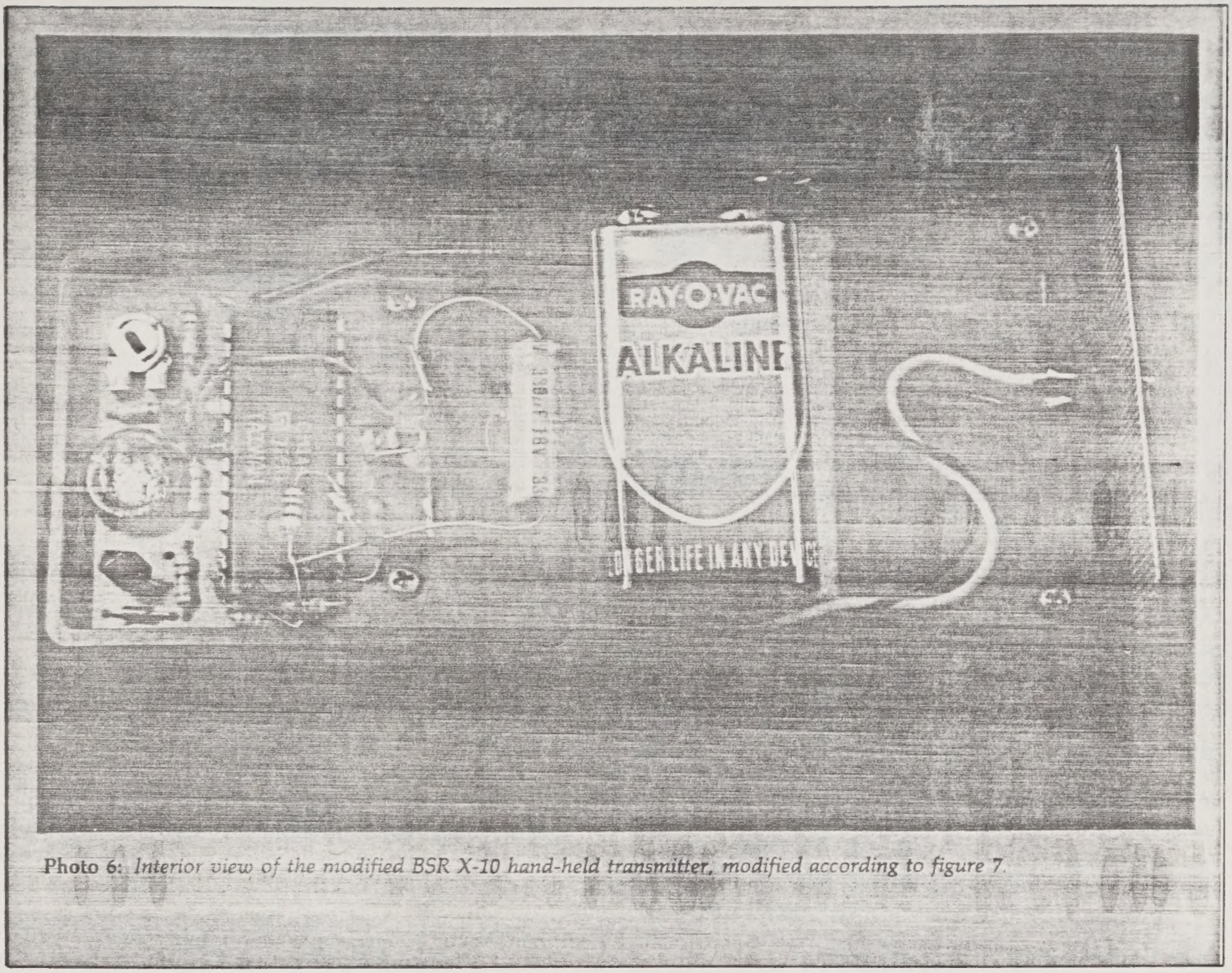


Photo 6: *Interior view of the modified BSR X-10 hand-held transmitter, modified according to figure 7.*

of the 40-kHz carrier will be marked by a high logic level coming out of the infrared receiver board for the duration of the 40-kHz signal. The coded data is sent as a series of 1.2-ms and 4-ms bursts, representing logic 0 and 1 respectively. The complete 13-bit message consists of a start bit (logic 1), 5 data bits corresponding to the key being pressed, 5 bits representing the logical inversion of the first 5 data bits, and 2 stop bits (logic 1). The binary codes and the transmission sequence are shown graphically in figure 8 on page 49.

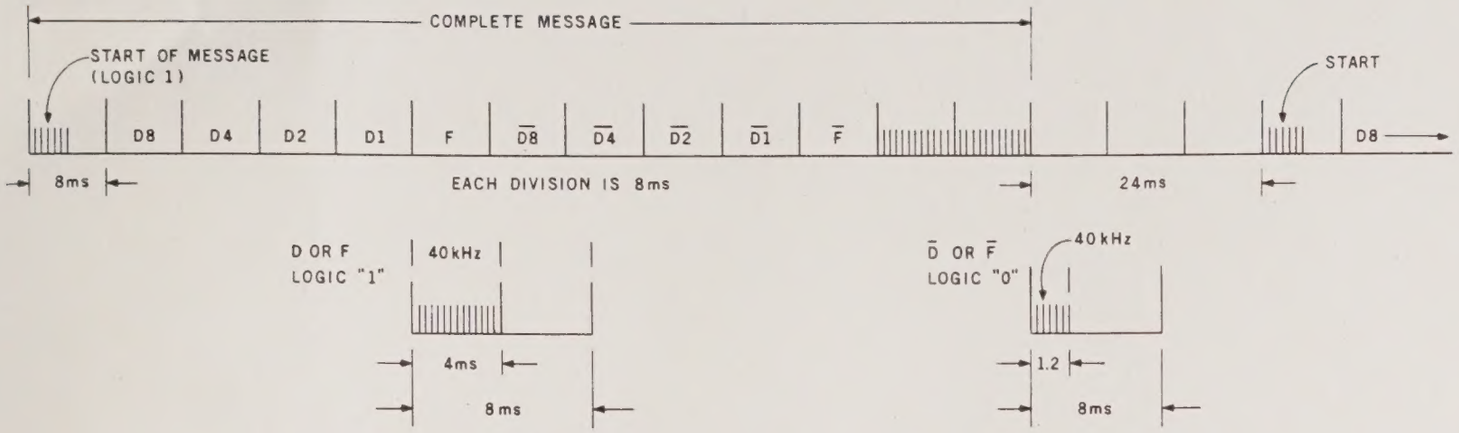
I'm sorry I was halfway through writing this article before I thought to use the BSR X-10 controller. After the idea struck home, I took a pizza break. Then I came back to the Circuit Cellar, added the modification of figure 7, and verified correct data reception using an oscilloscope. I haven't actually written the code to interpret the BSR controller's signals yet. But considering the well-documented transmission protocol used in the BSR, it may be easier to use this unit rather than figure out the unknown coding of a TV remote-control unit.

The longer the wavelength of its light, the higher the efficiency of an LED.

In Conclusion

Building a reliable infrared receiver has been a goal of mine for a long time. Many of my first designs did eventually work, but they couldn't be easily duplicated. Since I believe that many other experimenters are equally interested in IR communication and have experienced similar frustrations, I have arranged with Micromint to make available a complete kit of the infrared-communication circuits shown in figures 4 and 5. Included with these kits is a complete data sheet on the SN76832AN should you care to configure it for another frequency range.

If you try infrared communication and are successful, you might develop applications for it that have previously been ignored. Certainly experimenters like myself have been looking for better types of man/machine interaction than presently exist. Until computer speech recognition becomes



CHANNEL NUMBER OR FUNCTION	BINARY CODE D8	D4	D2	D1	F
1	0	1	1	0	0
2	1	1	1	0	0
3	0	0	1	0	0
4	1	0	1	0	0
5	0	0	0	1	0
6	1	0	0	1	0
7	0	1	0	1	0
8	1	1	0	1	0
9	0	1	1	1	0
10	1	1	1	1	0
11	0	0	1	1	0
12	1	0	1	1	0
13	0	0	0	0	0
14	1	0	0	0	0
15	0	1	0	0	0
16	1	1	0	0	0
ALL OFF	0	0	0	0	1
ALL LIGHTS ON	0	0	0	1	1
ON	0	0	1	0	1
OFF	0	0	1	1	1
DIM	0	1	0	0	1
BRIGHT	0	1	0	1	1

Figure 8: *Timing diagram and binary command-code table used by the BSR X-10 Home Control System.*

practical, we shall have to be satisfied with pushing buttons to communicate with computers. But perhaps the infrared transmitter and receiver in this article can make the connection a little more convenient.

Next Month:

It's been a long time since I wrote about time, time as measured by a computer's real-time clock. Next month, we'll look at plans for connecting a versatile clock to a personal computer. ■

Editor's Note: *Steve often refers to previous Circuit Cellar articles as reference material for each month's current article. Most of these past articles are available in reprint books from BYTE Books, 70 Main St., Peterborough, NH 03458.* Ciarcia's Circuit Cellar, Volume I *covers articles that appeared in BYTE from September 1977 through November 1978.* Ciarcia's Circuit Cellar, Volume II *contains articles from December 1978 through June 1980.* Ciarcia's Circuit Cellar, Volume III *contains the articles that were published from July 1980 through December 1981.*

Parts Source

The following is available from:

The Micromint Inc.
917 Midway
Woodmere, NY 11598
telephone (516) 374-6793
(for technical data)
(800) 645-3479
(orders only)

Infrared Transmitter/Receiver Kit $42

Includes two printed-circuit boards (one for the transmitter and one for the receiver) and all components shown in figures 4 and 5. Does not include the cases and power supplies shown in the prototype photos. Assembly manual and specification sheets provided.

Please include $2 for delivery within the United States or $6 for foreign delivery. Residents of New York state please add 7% sales tax.

References

1. Ciarcia, Steve. "Computerize a Home," BYTE, January 1980, page 28.
2. Ciarcia, Steve. "Self-Refreshing LED Graphics Display," BYTE, October 1979, page 58. Reprinted in *Ciarcia's Circuit Cellar, Volume II.* Peterborough, NH: BYTE Books, 1981, page 109.
3. Cohen, Charles. "Blue-light LED joins red and green chip in package," *Electronics,* October 6, 1981, page 6E.
4. Hewlett-Packard Optoelectronics Division—Applications Engineering Staff. *Optoelectronics Applications Manual.* New York: McGraw-Hill Book Company, 1977.
5. Seippel, Robert G. *Optoelectronics.* Englewood Cliffs, NJ: Prentice-Hall, 1981.

Copyright 1982 by Steven A. Ciarcia.
All rights reserved.

To receive a complete list of Ciarcia's Circuit Cellar project kits available from the Micromint, circle 100 on the reader-service inquiry card at the back of the magazine.

Danny:
Leigh and I felt you'd want to know what Terrapin is doing these days. Incidentally, we really sent this to this guy — Pat

6 January 1982

Terrapin, Inc.
678 Massachusetts Avenue
Room 205
Cambridge, MA 02139

Gentlemen:

SRI International is currently under contract to the U.S. Department of Defense to study possible applications of Artificial Intelligence and Robotics systems to military functions. Our interests include combat, combat support, logistics, and industrial/manufacturing functions, death and destruction.

Please send me descriptive information and cost data on any of your products which you feel might be applicable to these uses. It is important that the information include, if possible, your estimates of (1) the function(s) the system might perform, (2) system unit procurement cost (in quantities ranging from one to 10,000), (3) installation cost (if any) per unit, (4) annual cost of expendable supplies and spares per unit, (5) annual man-hours of labor required for operation and maintenance of one unit, and (6) system characteristics such as weight, size, power input, performance capacities, etc., with pictures.

The information you provide will be used only in SRI's analytical studies, and will not be considered an official offer by your company to supply the system(s) at the quoted prices.

Please call me at (415) 859-2592 if you have any questions.

Sincerely yours,

William Schubert

William Schubert
Senior Systems Analyst
Center for Defense Analysis

WS/gjn

SRI International
333 Ravenswood Ave. • Menlo Park, CA 94025 • (415) 326-6200 • Cable: SRI INTL MNP • TWX: 910-373-1246

Terrapin, Inc.

678 Massachusetts Avenue #205
Cambridge, Massachusetts 02139
617-492-8816

3:05am Tuesday, 19 January 1982

William Schubert
Center for Defense Analysis
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

Dear sir:

We must admit to some initial puzzlement at receipt of your communique of 6 January regarding possible applications of our robotics and artificial intelligence products to military functions.

However, always eager to contribute to the defense of our country from the ever-present threat, we put our best minds right to work on the problem and put together the enclosed report. We hope it will aid in your analysis.

Please note that the information we are providing is to be used only in your analytical studies, and is <u>not</u> to be considered an official offer by Terrapin to supply the systems at the quoted prices.

Please call us at (617) 492-8816 if you have any questions.

Sincerely yours,

Patrick G. Sobalvarro

Leigh L. Klotz
Senior Software Engineers
Terrapin, Inc.

PGS,LLK/bbng

Introduction

At Terrapin, we feel that our two main products, the Terrapin Turtle (TM), and the Terrapin Logo Language[1] for the Apple II, bring together the fields of robotics and AI to provide hours of entertainment for the whole family. We are sure that an enlightened application of our products can uniquely impact the electronic battlefield of the future.

The Terrapin Turtle (TM) is a small, versatile robot that can perform any number of complex tasks under computer control. A powerful AI programming language is necessary to realize the full potential of this advanced device.

The Terrapin Logo Language, developed at the Massachusetts Institute of Technology Artificial Intelligence Laboratory's Logo Group, is ideal for this application. The Logo language is a close relative of Lisp, the language used most widely in AI research. It fills the bill quite handily!

---

1. Logo trademark under license from Bolt, Beranek, and Newman, Inc.

Descriptive Information

1. Functions the system might perform.

While somewhat limited in range,[1] Turtles show great promise as all-terrain, high-accuracy system with excellent survivability in hostile environments. The Turtle's low observability, low vulnerability to ECM, and its multidirectional sensing capabilities make it an ideal reconaissance vehicle. Its ability to perform complex terminal maneuvering while pushing a heavy payload makes it a superb delivery vehicle as well.

Survivability

The Turtle enjoys very low observability, due to a minimal radar cross-section and an almost non-existent infra-red signature. In addition, its ground-hugging characteristics maximize terrain masking, resulting in lowered target acquisition by most classes of SSM and ASM threats. Its sophisticated sensors generate virtually no radar signature, so antiradiation missiles are useless against it. Furthermore, the Turtle is completely invulnerable to high-altitude SAM attacks.

With its powerful motors, the Turtle has very high initial acceleration. It climbs 45-degree grades with ease. Its maneuverability more than compensates for its somewhat low cruising speed. The Turtle can make a 180-degree turn in less space than any military vehicle currently in use by US forces, ground, air, or sea. With minor modifications, a Turtle could be constructed that could double its cruising speed for a terminal "dash" capability that would greatly enhance survivability in the endgame.

Because of the Turtle's compactness, with proper camouflage, weapons that depend on human vision are ineffective against it, as ground troops will find it hard to spot even at very close range.

Even if a suitable counter were found to all these properties of

---

1. Extended-range Turtle Mark I variants are currently in testing, however. See the section on Range.

the Turtle, it seems dubious that an enemy could afford to deploy counterweapons in sufficient numbers to nullify the possibility of defense saturation in the event of an all-out Turtle attack. The low system cost of the Terrapin Turtle (TM) make it an economic way to defeat defensive systems of increasing technological sophistication.

Range

The Mark I, Mod 0 Turtle has an effective range of some 3-4 meters, depending on its winding count. Range is most severely limited by the Turtle's C3,[1] but this limitation is trivial by comparison with the inherent advantages of wire-guidance. These advantages are discussed further below (see section Guidance).

Furthermore, our research department is currently engaged in the testing of a 100-mile C3 for the Turtle. The thrust of this research is towards the development of an Extended-Range Turtle II (TM). While this does result in a shorter tooth-to-tail ratio, we feel it could significantly enhance the battlefield capabilities of Turtle installations.

Guidance

The Terrapin Turtle (TM), like many missile systems in use today, is wire-guided. It has the wire-guided missile's robustness with respect to ECM, and, unlike beam-riding missiles, or most active-homing systems, it has no radar signature to invite enemy missiles to home in on it or its launch platform.

However, the Turtle does not suffer from that bugaboo of wire-guided missiles, i.e., the lack of a fire-and-forget capability. Often ground troops are reluctant to use wire-guided antitank weapons because of the need for line-of-sight contact with the target until interception is accomplished. The Turtle requires no such human guidance; once the computer controlling it

---

1. Computer Control Cable, often familiarly called the Turtle's "tail" in civilian applications.

has been programmed, the Turtle performs its mission without the need of human intervention. Ground troops are left free to scramble for cover.

But why stop there? Even more interesting scenarios are readily envisionable.

Because the Terrapin Turtle (TM) is computer-controlled, military data processing technicians can write arbitrarily baroque programs that will cause it to do pretty much unpredictable things. Even if an enemy had access to the programs that guided a Turtle Task Team (TM), it is quite likely that they would find them impossible to understand, especially if they were written in ADA. In addition, with judicious use of the Turtle's touch sensors, one could, theoretically, program a large group of turtles to simulate Brownian motion. The enemy would hardly attempt to predict the paths of some 10,000 turtles bumping into each other more or less randomly on their way to performing their mission. Furthermore, we believe that the spectacle would have a demoralizing effect on enemy ground troops.

Since the Terrapin Logo Language incorporates list structure, it is conceivable that someone will someday write an intelligent general problem-solving program in it. If this happened, truly intelligent guidance of Turtles would be possible, and since everyone knows that "smart" weapons are superior to "dumb" weapons, the Terrapin Turtle (TM) is superior to every other weapon developed to date.

Munitions

The Terrapin Turtle (TM) does not currently incorporate any munitions, but even civilian versions have a downward-defense capability. The Turtle can be programmed to attempt to run over enemy forces on recognizing them, and by raising and lowering its pen at about 10 cycles per second, puncture them to death.

Turtles can easily be programmed to push objects in a preferred direction. Given this capability, one can easily envision a Turtle discreetly nudging a hand grenade into an enemy camp, and then accelerating quickly away. With the development of ever smaller fission devices, it does not seem unlikely that the Turtle could be used for delivery of tactical nuclear weapons.

General

When controlled from a properly programmed computer, the Turtle can solve mazes, push and follow blocks, and draw complex geometric figures. In military terms, this means it can travel through mine fields and repel advancing troops. Large quantities of Turtles could conceivably be used to simulate battle plans. (Terrapin, Inc. does not supply software for these applications.)

The Terrapin Turtle performs the following functions:

Forward <centimeters>
Back <centimeters>
Left <degrees>[1]
Right <degrees>[1]
Penup
Pendown
Eyeson
Eyesoff
Hornoff
Hornhi
Hornlow

---

1. Convertible to mils for military applications.

2. System Unit Procurement Cost

Item	Unit Cost	Quantity
Turtle, Assembled	$599.95	1
Turtle, Assembled	$637.17	259
Turtle, Assembled	$599.95	10,000
Turtle, Kit	$399.95	1
Turtle, Kit	$424.76	259
Turtle, Kit	$399.95	10,000
Apple II Interface	$199.95	1
Apple II Interface	$212.35	259
Apple II Interface	$199.95	10,000

3. Installation Cost

The Terrapin Turtle is designed for installation at no cost by children and elementary school teachers. We feel that military installation cost should be under $10,000/unit.

4. Annual Cost of expendable supplies and spares per unit.

Ball-point Pen refills $0.59 (one spare included)

In the rugged terrain of the battlefield, under rigorous load conditions, it may be necessary to occasionally replace the TurtleTires (TM). Due to fluctuations in the world rubber market, quotation of exact prices is not possible.

5. Annual man-hours of labor required for operation and maintenance of one unit.

Operation costs for a Terrapin Turtle (TM) can essentially be reduced to cost of programmer time. Terrapin, Inc., offers demonstration software with every Apple II interface. If the functions provided by this software are satisfactory for military purposes, then no man-hours of programmer time need be tallied.

In any case, we feel that it may not be fair to consider operation costs of single units, because we purposely designed the Terrapin Turtle (TM) in such a manner that <u>every</u> Terrapin

Turtle (TM) can execute the same functions as every other Terrapin Turtle (TM). Thus there are no problems of software incompatibility, and a single team of programmers can, theoretically, operate any number of Turtles.

The Turtle's high reliability components and rugged construction minimize its life-cycle costs.[1] We feel that this should contribute to a high weapons-on-target-per-dollar potential. However, the Turtle's C3 will sometimes require maintenance, and, occasionally, rotation. Careful programming of the Turtle can reduce the need for rotation significantly. The challenge, simply stated, is to develop sophisticated, yet flexible, routing performance boundaries so as to maximize near-zero winding counts.

With the careful design of programs with provisions for evaluation of field-operation and recertification test results, together with failure/discrepancy analysis and corrective action, we feel that it should be possible to optimally extend the scheduled maintenance intervals and actually effect performance exceeding specifications. It can be expected that with a conscientious implementation of such programs very little time need be spent in C3 maintenance.

This estimate was, of course, computed taking only peacetime sortie rates into account. Under wartime conditions, we feel that up to a six-to-one increase in sorties might be experienced, and thus operationally ready (O.R.) rates can be expected to decrease accordingly.

---

1. It has long been the contention of our public relations department that the Turtle maintains itself. We feel that, if this is true, with a little work, Turtles might be made to maintain other devices, such as tanks, helicopters, and aircraft carriers. Thus we feel that a negative figure might even be appropriate for maintenance costs.

6. System Characteristics

The Terrapin Turtle (TM) weighs approximately 32 ounces. It measures approximately 8" in diameter, and is about 6" tall. Its power requirements are 15 volts at 2 amps. In normal trim, its power output is about 27.4 watts; however, an enhanced Turtle with "dash" capability would necessarily have higher power input and output specifications.

Danny:
Leigh and I felt you'd want to know what Terrapin is doing these days. Incidentally, we really sent this to this guy — Pat

6 January 1982

Terrapin, Inc.
678 Massachusetts Avenue
Room 205
Cambridge, MA 02139

Gentlemen:

SRI International is currently under contract to the U.S. Department of Defense to study possible applications of Artificial Intelligence and Robotics systems to military functions. Our interests include combat, combat support, logistics, and industrial/manufacturing functions, death and destruction.

Please send me descriptive information and cost data on any of your products which you feel might be applicable to these uses. It is important that the information include, if possible, your estimates of (1) the function(s) the system might perform, (2) system unit procurement cost (in quantities ranging from one to 10,000), (3) installation cost (if any) per unit, (4) annual cost of expendable supplies and spares per unit, (5) annual man-hours of labor required for operation and maintenance of one unit, and (6) system characteristics such as weight, size, power input, performance capacities, etc., with pictures.

The information you provide will be used only in SRI's analytical studies, and will not be considered an official offer by your company to supply the system(s) at the quoted prices.

Please call me at (415) 859-2592 if you have any questions.

Sincerely yours,

William Schubert

William Schubert
Senior Systems Analyst
Center for Defense Analysis

WS/gjn

SRI International
333 Ravenswood Ave. • Menlo Park, CA 94025 • (415) 326-6200 • Cable: SRI INTL MNP • TWX: 910-373-1246

Terrapin, Inc.

678 Massachusetts Avenue #205
Cambridge, Massachusetts 02139
617-492-8816

3:05am Tuesday, 19 January 1982

William Schubert
Center for Defense Analysis
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

Dear sir:

We must admit to some initial puzzlement at receipt of your communique of 6 January regarding possible applications of our robotics and artificial intelligence products to military functions.

However, always eager to contribute to the defense of our country from the ever-present threat, we put our best minds right to work on the problem and put together the enclosed report. We hope it will aid in your analysis.

Please note that the information we are providing is to be used only in your analytical studies, and is <u>not</u> to be considered an official offer by Terrapin to supply the systems at the quoted prices.

Please call us at (617) 492-8816 if you have any questions.

Sincerely yours,

Patrick G. Sobalvarro

Leigh L. Klotz
Senior Software Engineers
Terrapin, Inc.

PGS,LLK/bbng

Introduction

At Terrapin, we feel that our two main products, the Terrapin Turtle (TM), and the Terrapin Logo Language[1] for the Apple II, bring together the fields of robotics and AI to provide hours of entertainment for the whole family. We are sure that an enlightened application of our products can uniquely impact the electronic battlefield of the future.

The Terrapin Turtle (TM) is a small, versatile robot that can perform any number of complex tasks under computer control. A powerful AI programming language is necessary to realize the full potential of this advanced device.

The Terrapin Logo Language, developed at the Massachusetts Institute of Technology Artificial Intelligence Laboratory's Logo Group, is ideal for this application. The Logo language is a close relative of Lisp, the language used most widely in AI research. It fills the bill quite handily!

---

1. Logo trademark under license from Bolt, Beranek, and Newman, Inc.

Descriptive Information

1. Functions the system might perform.

While somewhat limited in range,[1] Turtles show great promise as all-terrain, high-accuracy system with excellent survivability in hostile environments. The Turtle's low observability, low vulnerability to ECM, and its multidirectional sensing capabilities make it an ideal reconaissance vehicle. Its ability to perform complex terminal maneuvering while pushing a heavy payload makes it a superb delivery vehicle as well.

Survivability

The Turtle enjoys very low observability, due to a minimal radar cross-section and an almost non-existent infra-red signature. In addition, its ground-hugging characteristics maximize terrain masking, resulting in lowered target acquisition by most classes of SSM and ASM threats. Its sophisticated sensors generate virtually no radar signature, so antiradiation missiles are useless against it. Furthermore, the Turtle is completely invulnerable to high-altitude SAM attacks.

With its powerful motors, the Turtle has very high initial acceleration. It climbs 45-degree grades with ease. Its maneuverability more than compensates for its somewhat low cruising speed. The Turtle can make a 180-degree turn in less space than any military vehicle currently in use by US forces, ground, air, or sea. With minor modifications, a Turtle could be constructed that could double its cruising speed for a terminal "dash" capability that would greatly enhance survivability in the endgame.

Because of the Turtle's compactness, with proper camouflage, weapons that depend on human vision are ineffective against it, as ground troops will find it hard to spot even at very close range.

Even if a suitable counter were found to all these properties of

---

1. Extended-range Turtle Mark I variants are currently in testing, however. See the section on Range.

the Turtle, it seems dubious that an enemy could afford to deploy counterweapons in sufficient numbers to nullify the possibility of defense saturation in the event of an all-out Turtle attack. The low system cost of the Terrapin Turtle (TM) make it an economic way to defeat defensive systems of increasing technological sophistication.

Range

The Mark I, Mod 0 Turtle has an effective range of some 3-4 meters, depending on its winding count. Range is most severely limited by the Turtle's C3,[1] but this limitation is trivial by comparison with the inherent advantages of wire-guidance. These advantages are discussed further below (see section Guidance).

Furthermore, our research department is currently engaged in the testing of a 100-mile C3 for the Turtle. The thrust of this research is towards the development of an Extended-Range Turtle II (TM). While this does result in a shorter tooth-to-tail ratio, we feel it could significantly enhance the battlefield capabilities of Turtle installations.

Guidance

The Terrapin Turtle (TM), like many missile systems in use today, is wire-guided. It has the wire-guided missile's robustness with respect to ECM, and, unlike beam-riding missiles, or most active-homing systems, it has no radar signature to invite enemy missiles to home in on it or its launch platform.

However, the Turtle does not suffer from that bugaboo of wire-guided missiles, i.e., the lack of a fire-and-forget capability. Often ground troops are reluctant to use wire-guided antitank weapons because of the need for line-of-sight contact with the target until interception is accomplished. The Turtle requires no such human guidance; once the computer controlling it

---

1. Computer Control Cable, often familiarly called the Turtle's "tail" in civilian applications.

has been programmed, the Turtle performs its mission without the need of human intervention. Ground troops are left free to scramble for cover.

But why stop there? Even more interesting scenarios are readily envisionable.

Because the Terrapin Turtle (TM) is computer-controlled, military data processing technicians can write arbitrarily baroque programs that will cause it to do pretty much unpredictable things. Even if an enemy had access to the programs that guided a Turtle Task Team (TM), it is quite likely that they would find them impossible to understand, especially if they were written in ADA. In addition, with judicious use of the Turtle's touch sensors, one could, theoretically, program a large group of turtles to simulate Brownian motion. The enemy would hardly attempt to predict the paths of some 10,000 turtles bumping into each other more or less randomly on their way to performing their mission. Furthermore, we believe that the spectacle would have a demoralizing effect on enemy ground troops.

Since the Terrapin Logo Language incorporates list structure, it is conceivable that someone will someday write an intelligent general problem-solving program in it. If this happened, truly intelligent guidance of Turtles would be possible, and since everyone knows that "smart" weapons are superior to "dumb" weapons, the Terrapin Turtle (TM) is superior to every other weapon developed to date.

Munitions

The Terrapin Turtle (TM) does not currently incorporate any munitions, but even civilian versions have a downward-defense capability. The Turtle can be programmed to attempt to run over enemy forces on recognizing them, and by raising and lowering its pen at about 10 cycles per second, puncture them to death.

Turtles can easily be programmed to push objects in a preferred direction. Given this capability, one can easily envision a Turtle discreetly nudging a hand grenade into an enemy camp, and then accelerating quickly away. With the development of ever smaller fission devices, it does not seem unlikely that the Turtle could be used for delivery of tactical nuclear weapons.

General

When controlled from a properly programmed computer, the Turtle can solve mazes, push and follow blocks, and draw complex geometric figures. In military terms. this means it can travel through mine fields and repel advancing troops. Large quantities of Turtles could conceivably be used to simulate battle plans. (Terrapin, Inc. does not supply software for these applications.)

The Terrapin Turtle performs the following functions:

Forward <centimeters>
Back <centimeters>
Left <degrees>[1]
Right <degrees>[1]
Penup
Pendown
Eyeson
Eyesoff
Hornoff
Hornhi
Hornlow

---

1. Convertible to mils for military applications.

2. System Unit Procurement Cost

Item	Unit Cost	Quantity
Turtle, Assembled	$599.95	1
Turtle, Assembled	$637.17	259
Turtle, Assembled	$599.95	10,000
Turtle, Kit	$399.95	1
Turtle, Kit	$424.76	259
Turtle, Kit	$399.95	10,000
Apple II Interface	$199.95	1
Apple II Interface	$212.35	259
Apple II Interface	$199.95	10,000

3. Installation Cost

The Terrapin Turtle is designed for installation at no cost by children and elementary school teachers. We feel that military installation cost should be under $10,000/unit.

4. Annual Cost of expendable supplies and spares per unit.

Ball-point Pen refills $0.59 (one spare included)

In the rugged terrain of the battlefield, under rigorous load conditions, it may be necessary to occasionally replace the TurtleTires (TM). Due to fluctuations in the world rubber market, quotation of exact prices is not possible.

5. Annual man-hours of labor required for operation and maintenance of one unit.

Operation costs for a Terrapin Turtle (TM) can essentially be reduced to cost of programmer time. Terrapin, Inc., offers demonstration software with every Apple II interface. If the functions provided by this software are satisfactory for military purposes, then no man-hours of programmer time need be tallied.

In any case, we feel that it may not be fair to consider operation costs of single units, because we purposely designed the Terrapin Turtle (TM) in such a manner that <u>every</u> Terrapin

Turtle (TM) can execute the same functions as every other Terrapin Turtle (TM). Thus there are no problems of software incompatibility, and a single team of programmers can, theoretically, operate any number of Turtles.

The Turtle's high reliability components and rugged construction minimize its life-cycle costs.[1] We feel that this should contribute to a high weapons-on-target-per-dollar potential. However, the Turtle's C3 will sometimes require maintenance, and, occasionally, rotation. Careful programming of the Turtle can reduce the need for rotation significantly. The challenge, simply stated, is to develop sophisticated, yet flexible, routing performance boundaries so as to maximize near-zero winding counts.

With the careful design of programs with provisions for evaluation of field-operation and recertification test results, together with failure/discrepancy analysis and corrective action, we feel that it should be possible to optimally extend the scheduled maintenance intervals and actually effect performance exceeding specifications. It can be expected that with a conscientious implementation of such programs very little time need be spent in C3 maintenance.

This estimate was, of course, computed taking only peacetime sortie rates into account. Under wartime conditions, we feel that up to a six-to-one increase in sorties might be experienced, and thus operationally ready (O.R.) rates can be expected to decrease accordingly.

---

1. It has long been the contention of our public relations department that the Turtle maintains itself. We feel that, if this is true, with a little work, Turtles might be made to maintain other devices, such as tanks, helicopters, and aircraft carriers. Thus we feel that a negative figure might even be appropriate for maintenance costs.

6. System Characteristics

The Terrapin Turtle (TM) weighs approximately 32 ounces. It measures approximately 8" in diameter, and is about 6" tall. Its power requirements are 15 volts at 2 amps. In normal trim, its power output is about 27.4 watts; however, an enhanced Turtle with "dash" capability would necessarily have higher power input and output specifications.

Approved I.R.S. Department of the Treasury—Internal Revenue Service 13-2678063 ☐ CORRECTED Form **1099-INT**

PAYER'S name, street address, city, state, and ZIP code		OMB No. 1545-0112 **1986** Statement for Recipients of	**Interest Income**
PARRAGUT MORTGAGE COMPANY 35 ENOREH STREET BELMONT MASSACHUSETTS 02178			
PAYER'S Federal identification number: 04-1612620	RECIPIENT'S identification number: 004-66-9313	1 Earnings from savings and loan associations, credit unions, bank deposits, bearer certificates of deposit, etc.: 21.22	**Copy B For Recipient**
RECIPIENT'S name, address, and ZIP code KARL SIMS 1000 BROADWAY SOMERVILLE MA 02143		2 Amount of forfeiture	3 U.S. Savings Bonds, etc.
		4 Federal income tax withheld: 0.00	
		5 Foreign tax paid (if eligible for foreign tax credit)	6 Foreign country or U.S. possession
Account number (optional) 333-0040257.7			

This is important tax information and is being furnished to the Internal Revenue Service. If you are required to file a return, a negligence penalty or other sanction will be imposed on you if this income is taxable and the IRS determines that it has not been reported.